Н.Б. Паршукова

ВИРТУАЛЬНАЯ РЕАЛЬНОСТЬ

Учебное пособие

Министерство просвещения Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Южно-Уральский государственный
гуманитарно-педагогический университет»

Н.Б. ПАРШУКОВА

ВИРТУАЛЬНАЯ РЕАЛЬНОСТЬ

УЧЕБНОЕ ПОСОБИЕ

Челябинск
2022

УДК 681.14 (021)
ББК 32.973.2–018я73 : 32.973.202я73
П 18

Паршукова, Н.Б. Виртуальная реальность: учебное пособие / Н.Б. Паршукова. – Челябинск: Изд-во Южно-Урал. гос. гуманит.-пед. ун-та, 2022. – 252 с.

ISBN 978-5-907611-48-1

Учебное пособие предназначено для обучения студентов основам технологии виртуальной реальности. Рассматриваются теоретические вопросы разработки VR, AR, создание и программирование 3D-графики, а также вопросы создания искусственного интеллекта в системах виртуальной реальности. Представлены лабораторные работы по программированию виртуальных миров в среде виртуального программирования Alice.

Пособие предназначено для обучающихся по направлениям 44.03.01 Педагогическое образование (профиль «Информатика»), 44.03.05 Педагогическое образование (профили «Математика. Информатика», «Физика. Информатика», «Информатика. Иностранный язык»), 09.03.02 Информационные системы и технологии, 44.04.01 Педагогическое образование (магистерская программа «Информатика и робототехника в образовании»). Может быть использовано для преподавания дисциплин «Виртуальная реальность», «Основы виртуальной реальности», «Виртуальные технологии в образовании».

Рецензенты

Т.В. Карпета, канд. физ.-мат. наук, ЮУрГУ

О.А. Дмитриева, канд. пед. наук, ЮУрГГПУ

ISBN 978-5-907611-48-1

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Быстрое развитие систем программирования, увеличение производительности микропроцессоров, разработка новых устройств для передачи информации позволило создать новый класс программно-аппаратного обеспечения, позволяющий не только наблюдать виртуальную среду, но и действовать в ней в интерактивном режиме. Такое программно-аппаратное обеспечение получило название VR-систем, или виртуальная реальность.

Виртуальная реальность (компьютерная модель реальности) – созданный техническими средствами мир (объекты и субъекты), передаваемый человеку (посетителю этого мира) через его ощущения: зрение, слух, обоняние и другие. Виртуальная реальность имитирует как воздействие, так и реакции на воздействие. Для создания убедительного комплекса ощущений реальности компьютерный синтез свойств и реакций виртуальной реальности производится в реальном времени.

VR-системы широко применяются при обучении. Отработка умений и навыков в виртуальной среде с заранее заданными параметрами, зачастую более сложными, чем в реальности, позволяет готовить качественно и в достаточно короткий срок специалистов для управления транспортными средствами, летательными аппаратами, роботизированной техникой; специалистов для ликвидации последствий стихийных бедствий, для предотвращения террористических атак, специалистов в области хирургии; виртуальные системы помогают оттачивать сложные технические операции в нефтяной, металлургической, химической промышленности и др. Подобные системы широко применяются в транспортной промышленности. Они значительно

экономят средства предприятий при разработке прототипов новых транспортных средств, при проведении испытаний конструкций на прочность (краш-тесты).

Большой интерес к созданию VR-систем во всем мире породил масштабные исследования в области компьютерных наук, математики, физики, психологии, биологии, лингвистике. Многие идеи, первоначально направленные на применение в VR-системах, были реализованы и в других областях (виртуальная реклама, умные доски, интерфейсы для управления без каких-либо дополнительных устройств, логистика и многое другое).

В этой связи существенным вкладом в подготовку будущего учителя информатики и инженера-программиста [34] является обобщение целого ряда дисциплин, реализуемое в курсе «Виртуальная реальность». В настоящее время специальной литературы по данному курсу представлено недостаточно.

Данное учебное пособие может быть полезно студентам бакалаврам направлений 09.03.02 Информационные системы и технологии, 44.03.01 Педагогическое образование (профиль «Информатика») и 44.03.05 Педагогическое образование (профили «Математика. Информатика», «Физика. Информатика», «Информатика. Иностранный язык»), магистрантам по направлению 44.04.01 Педагогическое образование (магистерская программа «Информатика и робототехника в образовании»), может быть использовано в качестве учебной литературы курсов дисциплинам «Виртуальная реальность», «Основы виртуальной реальности», «Виртуальные технологии в образовании», а также в качестве источника для научно-исследовательских работ.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ О ТЕХНОЛОГИИ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ

1.1. Термин «виртуальная реальность». Причины появления виртуальной реальности

1.1.1. Генезис термина «виртуальная реальность»

Появившиеся в последнее десятилетие технологии сделали популярными два новых понятия: виртуальная реальность и киберпространство. Под понятием «виртуальная реальность» (искусственная реальность, электронная реальность, компьютерная модель реальности) (от лат. *virtus* – потенциальный, возможный и лат. *realis* – действительный, существующий; англ. *virtual reality*, VR) понимается созданный техническими средствами мир (объекты и субъекты), который передается человеку (посетителю этого мира) через его ощущения: зрение, слух, обоняние и другие. Виртуальная реальность имитирует как воздействие, так и реакции на воздействие. Для создания убедительного комплекса ощущений реальности компьютерный синтез свойств и реакций виртуальной реальности производится в реальном времени.

Надо признать, что идеи создания искусственной реальности занимали умы писателей, философов и ученых задолго до появления компьютеров.

В конце XVII в. барон Готтфрид Вильгельм фон Лейбниц, пытаясь найти универсальный язык понимания природы, открыл формальную логику. Он создал символьный язык, описав с помощью которого любое явление, можно делать выводы в независимости от того, что именно представлено данными символами. Универсальный набор символов (*characteristica universalis)* мог описать любое человеческое знание и не был ограничен материальным существованием знания. «Позвольте нам загрузить все это в нашу общую систему, а затем просто сесть и вычислить», – говорил Лейбниц.

Пытаясь понять механизмы интуиции человеческого разума процесс, Лейбниц разработал метафизическую теорию, получившую название монадологии, теорию реальности, описываемую системой монад.

Вселенная монад – точное философское описание Всемирной компьютерной сети.

Сеть – это центральная бесконечная монада, объединяющая и координирующая множество пользователей – монад, отображающих целый мир через свои терминалы.

Никто из пользователей реально не видит Сеть в целом, но, увлекаемый своими собственными побуждениями, он отображает ее часть через свой терминал, изучая и достраивая эту виртуальную вселенную. Сеть, аналогично божественному мышлению, позволяет мгновенно достигать требуемого решения – информации, для получения которой в реальном «физическом» пространстве необходимо совершить немало шагов [12].

Чрезвычайно полезным для понимания виртуальной реальности стало утверждение Канта о том, что идеи пространства и времени известны человеку прежде восприятия. Дело в том, эти два понятия, в будущем рассматриваемые Эйнштейном и Минковским как единое целое, формируют базис, на основе которого зиждется любое понятие реальности. Мы не можем говорить о реальности без времени и не можем представить реальность без

пространства. Тем самым, рассматривая реальность, в которой может существует человек, и тем более искусственную реальность, мы должны заранее определить понятия пространства и времени [12].

Научная фантастика, жанр научно-популярной литературы, в середине XX в. переживал бурный подъем. Произведения Рея Брэдбери, Маршалла Маклюэна, С. Лема и многих других авторов содержат описания искусственно создаваемых человеком миров [11, 29, 31]. Авторы не только увлекательно описывают искусственные миры, они затрагивают философские и психологические проблемы влияния этих миров на человечество.

а) б) в)

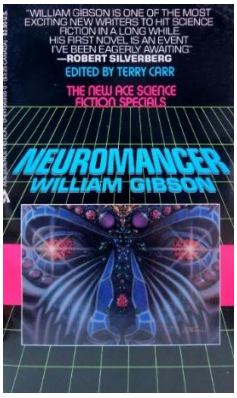
Рис. 1. Р. Брэдбери (а); иллюстрация к рассказу «Вельд» (б); проект CAVE (в)

В середине XX в. Рей Брэдбери написал свое произведение «Вельд» [11]. В нем описывалась семья, живущая в доме с детской комнатой, в которой за счет «телевизионных стен» и другого оборудования достигалась имитация другой реальности. Детям очень нравится эта комната, но в какой-то момент родителям становится ясно, что комната воспитывает жестокость в детях, дети постоянно вызывают реальность африканского вельда. Родители решают выключить комнату. В отместку разгневанные

дети оставляют родителей в детской, где материализовавшиеся из виртуальной реальности львы съедают родителей. Самое интересное, что подобного рода система с отображением на стены, правда без возможности материализации львов, была действительно создана в 1996 г. и называлась CAVE (пещера).

Слово «киберпространство» (cyberspace), часто ассоциируемое с понятием «виртуальная реальность» впервые было упомянуто фантастом Уильямом Гибсоном в 1984 г. в его романе Neuromancer (Нейромант) [14] для обозначения глобальной многопользовательской виртуальной среды. Вопросы, поднятые Гибсоном, достаточно полно отражают идею создаваемых искусственно виртуальных миров, и поэтому ссылки на его произведения можно встретить в большинстве работ, связанных с исследованием виртуальной реальности. Уильям Гибсон детально описал киберпространство, он описал тот искусственный мир, к которому сейчас стремятся ученые, занимающиеся компьютерной графикой и интерфейсами «человек – компьютер». В произведении «Нейромант» рассмотрены такие понятия, как искусственный интеллект, виртуальная реальность, генная инженерия, транснациональные корпорации, киберпространство (компьютерная сеть, матрица) задолго до того, как эти понятия стали популярными в массовой культуре.

а)



б)

Рис. 2. У. Гибсон (а); обложка первого издания «Нейромант» (б)

Это произведение очень интересно тем, что в нем поднят вопрос о влиянии средств массовой информации на индивида задолго до того, как обозначенная проблема была освещена на глобальном уровне в конце XX столетия. Насколько опасен мощнейший информационный поток, порождаемый современными информационными системами? Насколько его можно контролировать? Насколько он влияет на человека, и прежде всего на ребенка с еще не окрепшей психикой? Насколько медиаобразование может справиться с этой проблемой? Все эти вопросы пока не получили окончательного ответа.

В 1964 г. Маршалл Маклюэн представил свою работу «Understanding Media» («Понимание медиа» [31]). Эта работа являлась революционной и была интересна тем, что впервые автор пытался рассмотреть процесс влияния средств передачи информации на общество. Идеи, представленные в этой книге, впервые описали общество, которое в дальнейшем будут называть информационным.

Рис. 3. Маршалл Маклюэн

Одна из концепций данной книги выражена фраза «информационная среда – это сообщение». Под этой фразой Маклюэн подразумевал несколько идей. Во-первых, различные информационные среды предназначены для различных сообщений. Телевидение, например, предназначено

для передачи кратких видео- и звуковых образов и непригодно для передачи сложных аргументов и объяснений. Во-вторых, информационная среда сама своим существованием передает какой-то вид сообщения. Одни информационные среды считаются «холодными», а другие – «горячими». Сама среда имеет свой характер и смысл. В-третьих, информационная среда является сообщением с той точки зрения, что новые средства коммуникации влияют на процесс человеческого общения. Такие средства передачи информации, как телевидение, покрывая своим влиянием несравненно большие пространства, чем было возможно ранее, создают «глобальную деревню».

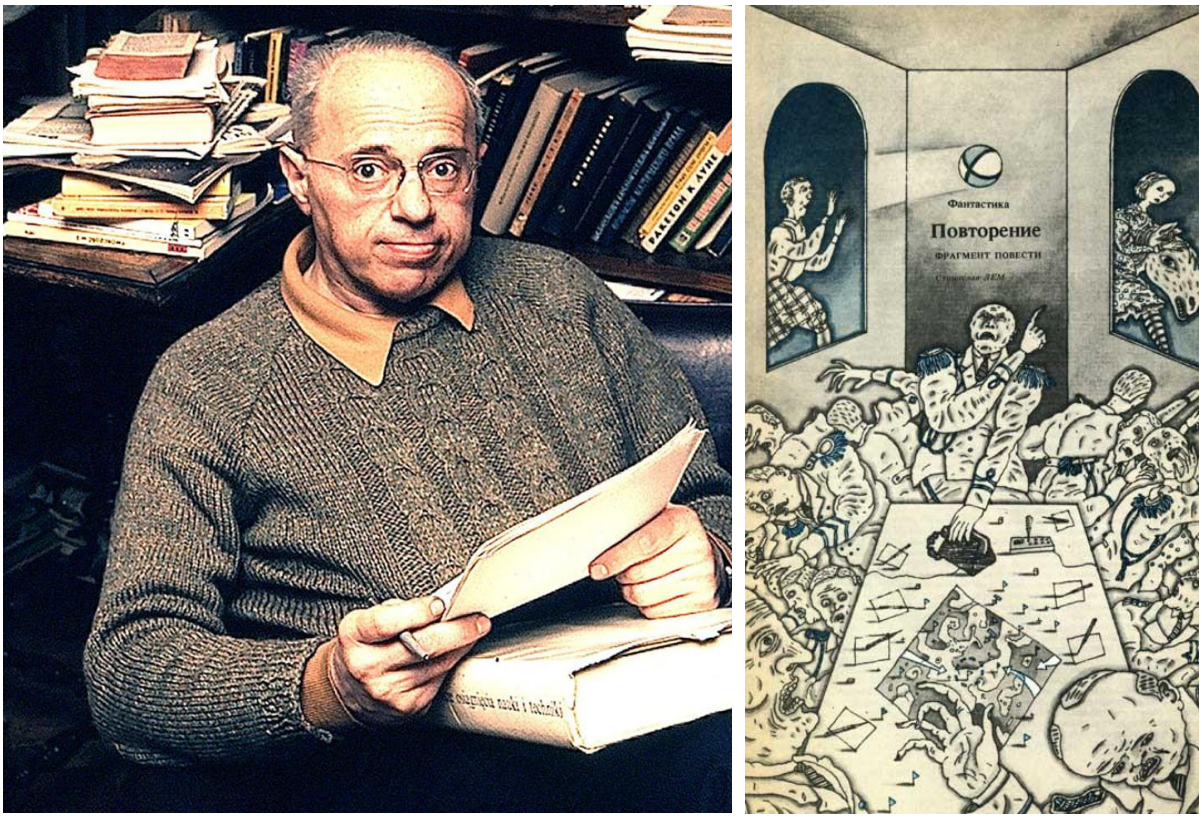
Понятие «глобальная деревня», появившееся в научных кругах в середине XX столетия, благодаря Маршаллу Маклюэну, применяется в основном метафорически, давая описание Всемирной информационной сети. В этой сети расстояния между людьми перестают иметь какое-либо значение для коммуникации, время и пространство словно стираются, и вместе с тем сближаются культуры, традиции, мировоззрения и ценности. Из-за большой скорости обмена информацией у человека появилось преимущество: он может стремительно реагировать на происходящее в мире, принимать и распространять информацию.

Только через пять лет, 27 октября 1969 г. два компьютера, удаленные друг от друга, смогли обменяться сообщениями через коммутируемую телефонную линию, положив начало глобальной телекоммуникации. Это был первый шаг в развитии охватившей сегодня весь земной шар сети Интернет, первый действительно реальный шаг к «глобальной деревне» Маклюэна. Описанные им процессы стали действительно происходить в гораздо большем масштабе, чем он предполагал. Интернет действительно стал самим сообщением, он стал влиять на методы общения людей, значительно потеснив такие средства коммуникации, как почта. Он предоставил единый метод доступа к любой информации, находящейся в мире.

Хотя технологии, существующие на сегодняшний день, едва ли справляются с тем, чтобы воссоздать видео- и аудиообразы, необходимые для систем виртуальной реальности, ситуация меняется не по дням, а по часам. Продвижение 3D-технологий уже сейчас позволяет создать более-менее реалистичное изображение, проблема с реалистичным звуком практически решена. Гораздо больше проблем возникает с передачей ощущений другими органами чувств. Хотя тактильные ощущения уже возможно передавать, технологии передачи тактильных ощущений находятся только в зачаточном развитии. Проблемы с передачей запаха и вкуса совсем не решены, так что «с виртуальным пирожком нам придется повременить».

Движение в виртуальном мире также является проблемой: естественная ходьба в виртуальном мире вынужденно будет сопровождаться движением в мире физическом. Конечно, можно попытаться передать перемещение в виртуальном пространстве через тренажер (беговой дорожке, или в «виртуальной сфере» братьев Латыповых [37]), но все равно такие ощущения не перенесут полностью в новый мир, оторванный от реального. Хотя и эти проблемы можно решить, подключившись, как у Гибсона, напрямую к центральной нервной системе и исключив тем самым использование естественных органов чувств. Однако не стоит забывать, что мы все равно останемся в реальном мире, который будет предъявлять на нас свои физические права.

Другой проблемой, встречающейся при создании виртуального мира, является то, что именно мы должны описать этот мир, его правила, его природу, его обитателей. С этой точки зрения очень интересно рассмотреть произведение Станислава Лема «Повторение» [29] и реально созданный и существовавший в период с июня 1986 г. по май 1988 г. проект LucasFilm «Habitat» («Среда обитания»). Эти две работы объединяет одно: обе они являются работами по «практической социологии».



а) б)

Рис. 4. С. Лем (а); иллюстрация к рассказу «Повторение» (б)

В произведении Лема ученые Трурль и Клапауций пытаются по указанию короля Ипполита Сармандского создать совершенный мир. Однако, создавая очередной, казалось бы, «идеальный» мир, они встречаются с различными проблемами. Поведение обитателей нового мира противоречит понятию совершенства. В проекте «Habitat» разработчики тоже встречаются с проблемами в созданном ими мире. Только источниками проблем являются не выдуманные, как у Лема, причины, а причины, реально существующие в созданном виртуальном мире. И так же, как у Лема, создатели «Habitat» каждый раз пытались решать проблемы, изменяя само виртуальное бытие.

Как ни странно, развитие событий и в «Повторении», и в «Habitat» закончилось практически одним и тем же результатом. В последних строках

«Повторения» Трурль показывает королю доказательство в «невозможности сотворения нового мира», говоря: «Я-то имел в виду совершенный!». Официально проект Habitat был свернут по финансовым причинам, но можно сказать, что он дошел до своего идеологического конца. «Habitat» был создан, в первую очередь, как средство в индустрии развлечений, и, как и всякая игра, исчерпал себя своей ограниченностью. Ни одна игра не может длиться чрезвычайно долго именно потому, что она имеет ограниченный набор правил, ограниченное пространство взаимодействия. Именно это отличает игру от жизни – рамки допустимых правил. И это же отличие накладывает свой отпечаток на все миры, создаваемые искусственно. Для реального мира характерно неограниченное пространство правил, создающее неограниченное пространство возможностей.

Рис. 5. Заставка к игре «Habitat»

В 1964 г. в Кракове вышла книга Станислава Лема «Summa technologie» («Сумма технологий») [29], в которой целая глава была посвящена фантомологии. По Лему, фантоматика – это область знания, решающая проблему создания действительности, которая для разумных существ, живущих в ней, ничем не отличалась бы от реальности, но подчинялась бы другим законам. Фантоматика предполагает создание двусторонних связей между «искусственной действительностью» и воспринимающим ее человеком.

Фантоматика предполагает создание такой ситуации, когда никаких выходов из созданного фиктивного мира в реальную действительность нет. Фантоматизация – это «короткое замыкание», то есть подключение человека к машине, фальсифицирующей действительность и изолирующей его от внешней среды. Эти формулировки фактически представляют собой прообраз современного определения виртуальной реальности. Виртуальная реальность – это компьютерная система, применяемая для создания искусственного мира, пользователь которой ощущает себя в этом мире, может управлять им, манипулировать его объектами. В произведениях С. Лема достаточно подробно описывается «антиглаз», укрепляемый на пользователе при помощи специальных очков (устройство ввода визуальной информации в глаз человека, то, что сейчас называется eyephone).

Вопросы, тем или иным образом имеющие отношение к виртуальной реальности, рассмотрены Лемом в различных аспектах и во многих других произведениях [35].

Впоследствии идея о виртуальных мирах была продолжена и в произведениях современных писателей, в частности, в романе С. Лукьяненко «Лабиринт отражений» [30].

1.1.2. История развития виртуальной реальности

Идеи, которые писатели-фантасты, футурологи вкладывали в свои труды, достаточно быстро нашли отражение в работах инженеров, математиков и программистов всего мира. Виртуальная реальность – мир, смоделированный с помощью компьютерных технологий, в который пользователь может погрузиться с помощью специальных сенсорных устройств. Технологии виртуальной реальности прошли огромный путь от первых экспериментов в 60-х годах XX в. до современных шлемов виртуальной реальности.

Благодаря тренажеру Link Trainer (1929 г.) у людей появилась возможность в процессе обучения моделировать реальные ситуации. Тренажер был предназначен для обучения пилотов и позволял соединить визуальное восприятие с восприятием движения и звука. Первоначальное применение данного аппарата предшествует изобретению компьютера. Link Trainer был летным тренажером, в исходной модели которого использовались движущаяся картинка и пневматические передачи, подобные органным трубам (рис. 6). Рычажный тренажер марки Link Trainer, запатентованный в 1929 г., заставлял моделирующее устройство двигаться, вращаться, падать, изменять курс и таким образом создавал удовлетворительное ощущение движения.

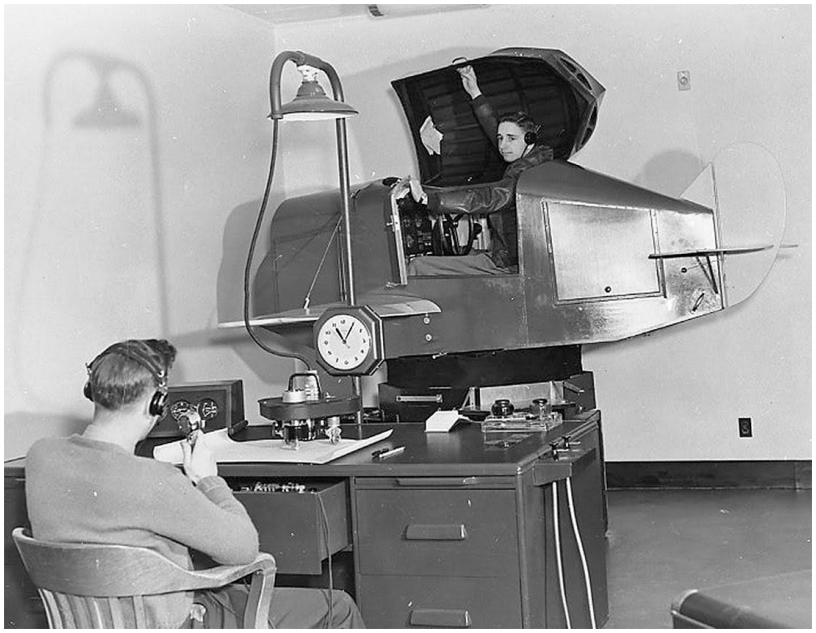


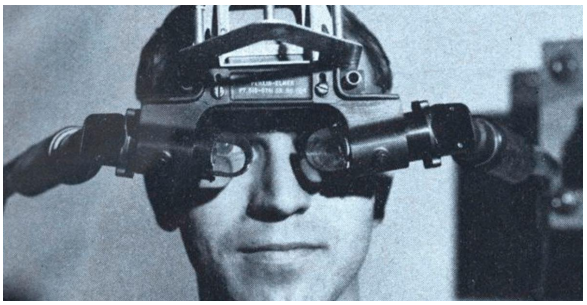
Рис. 6. Link Trainer

Первой системой виртуальной реальности принято считать прототип мультисенсорного симулятора, который был представлен публике в 1962 г. Разработчик Мортон Хейлинг называл свое устройство Сенсорама (Sensorama). Сенсорама погружала зрителя в виртуальную реальность при помощи коротких фильмов, которые сопровождались запахами, ветром (при помощи фена) и шумом мегаполиса с аудиозаписи (рис. 7, а). Спустя

5 лет Айвен Сазерленд описал и сконструировал первый шлем, изображение на который генерировалось при помощи компьютера. Шлем Сазерленда позволял изменять изображения соответственно движениям головы (зрительная обратная связь) [27].

В 1965 г. Сазерленд описал концепцию воздействия компьютерной имитации мира на пользователя через специальный шлем, который создаёт настолько реалистичную иллюзию, что человек не способен отличить имитацию от действительности, при этом у пользователя есть возможность взаимодействовать с объектами в виртуальной реальности.

Описанные в статье возможности в дальнейшем воспринимались исследователями и разработчиками как цель, концепция стала восприниматься как список будущих достижений и некий футуристический план. Пик развития технологии Сазерленд видел в том, чтобы пользователь мог сравнить опыт от использования виртуальной реальности с кэрроловским путешествием Алисы в Страну чудес.



а) б)

Рис. 7. Sensorama (а); устройство «Дамоклов меч» Айвена Сазерленда (б)

В 1968 г. Сазерленд совместно со своим учеником и коллегой Бобом Спроуллом разработали первый компьютерный шлем виртуальной реальности. Его назвали «Дамоклов меч» в связи с характерными особенностями стационарного крепления. Устройство было достаточно простым и отображало на экране только примитивные 3D-модели в виде незамысловатых объемных геометрические форм.

«Дамоклов меч» (рис. 7, б) был оснащен отслеживанием движений головы, в зависимости от которых менялась перспектива на экране. Устройство было исключительно лабораторным, в первую очередь из-за своей высокой массы, которая требовала крепление к потолку [27].

В 1970-х гг. компьютерная графика полностью заменила видеосъемку, до того использовавшуюся в симуляторах. Графика была крайне примитивной, однако важным было то, что тренажеры (это были симуляторы полетов) работали в режиме реального времени. Ещё одним, скорее идеологическим и концептуальным, нежели технологическим, прорывом стала фильм-карта Аспена (штат Колорадо), созданная командой исследователей из Массачусетского технологического университета (рис. 8).

Рис. 8. Виртуальная экскурсия по Аспену

В середине 1980-х гг. появились системы, в которых пользователь мог манипулировать трехмерными объектами на экране благодаря отклику на движения руки.

В 1989 г. виртуальная реальность была показана публике, тогда же закрепился сам термин «виртуальная реальность». Этот термин предложил Джарон Ланье (J. Lanier), определивший виртуальную реальность как «генерируемую компьютером, интерактивную, трехмерную среду, в которую погружается пользователь».

В 1985 г. появляется компания, оставившая заметный след в развитии виртуальной реальности, – VPL Research. Её основали Джарон Ланье и Томас Циммерман. VPL Research известна как первая компания, начавшая серийное производство и продажу шлемов и перчаток виртуальной реальности. В 80-х гг. XX в. компания разработала такие системы, как DataGlove, EyePhone HMD и Audio Sphere (рис. 9) [26].

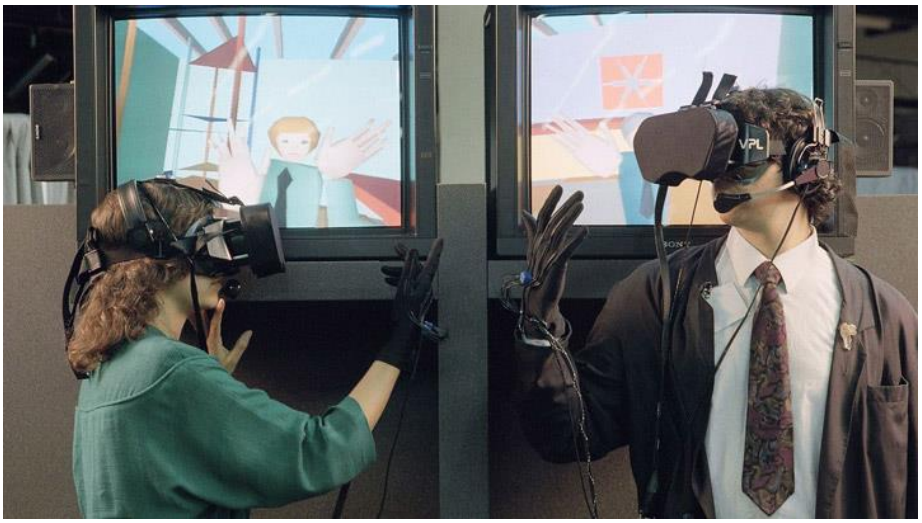


Рис. 9. Шлем и перчатки виртуальной реальности (VPL Research)

В 1989 г. Скотт Фостер создал компанию Crystal River Engineering Inc, которая приняла деятельное участие в проекте NASA. Национальное агентство использовало хорошо известные к тому моменту средства вирту-

альной реальности. Среди инноваций было применение полноценного костюма, а также перспективной по степени реалистичности аудиосистемы. Особенностью последней стало использование бинауральных эффектов вместо банального стерео, именно этой системой занималась компания Фостера (рис. 10).

Рис. 10. Устройства виртуальной реальности для NASA

В 1990-х г. стремительное развитие компьютерных технологий позволило совершенствовать параметры интерактивности; появилось сложное программное обеспечение и многочисленные исследовательские центры, разрабатывающие методы применения технологий виртуальной реальности в образовании, медицине, промышленности, военных и космических исследованиях.

С тех пор виртуальная реальность отождествляется с более глубоким подходом к созданию компьютерных моделей реальности, связанным со многими сложностями. Для нее нужны, как минимум, головной дисплей и перчаточное устройство (или другие средства управления виртуальными объектами). Полное погружение требует от пользователя применять сенсорный костюм, передающий данные о движениях в компьютер. Головной дисплей – это два очень маленьких видеомонитора, установленных так, что

каждый из них находится перед соответствующим глазом; человек смотрит на изображение через специальные широкоугольные линзы. Размещение этих устройств в маске или шлеме таково, что глаза могут принимать изображение, которое мозг идентифицирует как трехмерное. Некоторые дисплеи снабжены наушниками, создающими звуковую среду. Другие методы, как, например, специальные электронные очки, скорость изображения в которых сопоставима с видеодисплеями, позволяют пользователям работать в реальной среде, одновременно обращаясь к изображениям в среде виртуальной.

1.1.3. Отличие виртуальной реальности от других способов получения информации

Предшественником виртуальной реальности, несомненно, является телевидение. Фактически, оно уже давно используется массой людей для ухода в несуществующую, выдуманную реальность, в совершенно виртуальные события мыльных опер, телевизионных игр, мультсериалов или триллеров. При этом люди иногда оказываются втянутыми в придуманный мир настолько, что воспринимают телевизионных персонажей так, как если бы те были равноправными членами их семьи, сопереживают происходящим им, как близким людям. Такое восприятие не может не оказывать сильного воздействия на психику.

Телевидение, книги и другие средства массмедиа еще не могут считаться настоящей виртуальной реальностью, поскольку не обладают возможностью интерактивного взаимодействия. Другими словами, все перечисленные выше средства передачи информации не дают нам обратной связи: мы наблюдаем некое действие на экране телевизора, однако наша реакция никак не может повлиять на это действие, чтобы изменить его ход.

В этом смысле следующий, более глубокий уровень вовлеченности в виртуальность дают компьютерные игры. Хотя события, происходящие в игре, безусловно являются менее настоящими, чем те, что можно наблюдать на телеэкране, однако тот факт, что игрок непосредственно участвует в этих событиях, создает гораздо более мощный эффект погружения в виртуальный мир. Кроме того, исследования показали, что, будучи не совсем настоящим и оставляя простор для фантазии, такой мир часто становится более привлекательным, чем тот, который является фотографической копией реальной действительности.

Термин «киберпространство» является еще одной специфической разновидностью виртуальной реальности. На сегодняшний день глобальным всемирным киберпространством стала сеть Интернет, давшая обширное поле для применения виртуальных технологий. Такие технологии используются в работе виртуальных магазинов: пользователь получает на экране трехмерную модель комнаты магазина, где он может рассмотреть товары на полках, просто подойдя к соответствующей полке; щелкнув мышкой по соответствующему товару, он можете детально с ним ознакомиться; может взять его и подойти к кассе; щелкнув мышкой, может выбрать в меню «оплату» и тут же оплатить товар по кредитной карточке; и затем этот товар будет доставлен ему домой.

Виртуальные банки тоже используют технологию мультимедиа для создания реалистического компьютерного образа филиала. Такой филиал может быть доступен для пользователя, который не выходит из дома или из офиса. Виртуальный банк предоставляет целый ряд традиционных и нетрадиционных банковских услуг.

Наконец появился совершенно особый тип виртуальных организаций, не имеющих единого управляющего центра и четкой структуры, рассредо-

точенных по всей планете и юридически неидентифицируемых, однако часто оказывающихся достаточно работоспособными и эффективными благодаря внутреннему объединяющему моральному духу [15, 16].

1.1.4. Виды систем виртуальной реальности

Если следовать строгой терминологии, то все вышеперечисленное представляет собой только квази-VR, поскольку не обеспечивает полной вовлеченности человека в виртуальный мир с максимальным задействованием органов чувств. Полноценная VR-система должна обладать следующими свойствами: она отвечает на действия пользователя (интерактивность), в реальном времени представляет виртуальный мир в виде трехмерной графики и дает эффект погружения (рис. 11) [38].

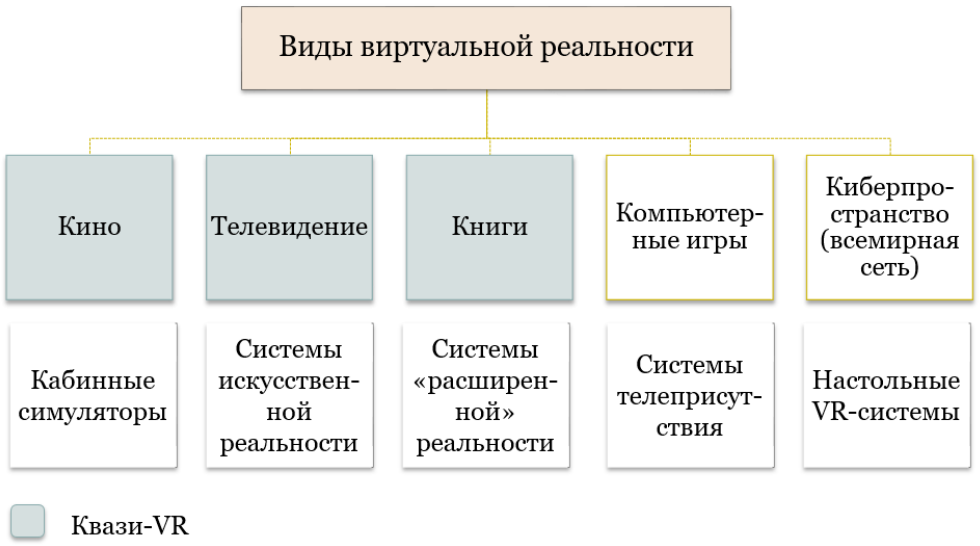


Рис. 11. Виды виртуальной реальности

Сегодня есть несколько типов более-менее массовых VR-систем.

Кабинные симуляторы (cab simulators) порождены авто- и авиа- тренажерами. В них пользователь садится в кабину и видит перед собой в окне

дисплей компьютера, на котором изображены некие ландшафты. Если пользователь начнет двигать управляющими ручками (рычагами или рулем), на дисплее будет соответственно изменяться ландшафт (рис. 12).

Рис. 12. Кабинный симулятор

Системы искусственной реальности (artificial, projected reality) дают возможность пользователям видеть реальные видеозаписи друг друга, встроенные в виртуальное пространство трехмерных образов. Эти системы не требуют головных дисплеев и могут успешно использоваться для непросвещенных пользователей. Идея совмещения видео и компьютерной графики в реальном времени породила, в частности, технологию виртуальных студий, при которой изображение на экране телевизора в реальном времени складывается из видеозаписей участников передачи (реально находящихся в пустой студии) и трехмерных миров, которые компьютер генерирует и соединяет с этой видеозаписью (рис. 13).

Рис. 13. Искусственная реальность на ТВ

Системы «расширенной», дополненной реальности (augmented reality, AR) представляют изображение на экране головного дисплея прозрачным, так что пользователь видит одновременно и свое реальное окружение, и виртуальные объекты, генерируемые компьютером на экране (рис. 14).

Рис. 14. Системы AR

Системы телеприсутствия (telepresence) используют видеокамеры и микрофоны для погружения в виртуальное окружение пользователя, который либо сморит в дисплей шлема, соединенный с подвижной камерой на платформе, либо управляет джойстиком без шлема. Такого рода системы были установлены на космическом корабле Pathfinder, который в июле

1997 г. «приземлился» на Марс. С помощью этих систем ученые могли рассматривать и фотографировать поверхность планеты (рис. 15).

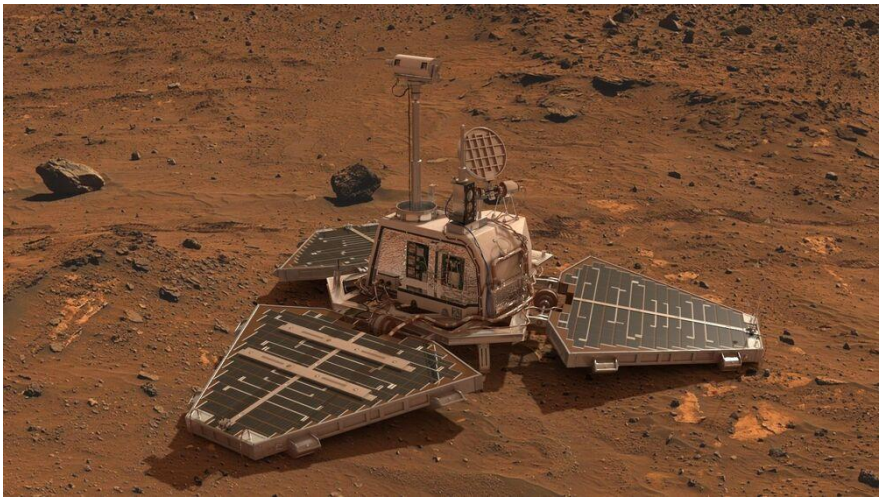


Рис. 15. Марсоход Pathfinder на телеуправлении

Настольные VR-системы (desktop VR) представляют виртуальную реальность с помощью больших мониторов или проекторов. Это хороший инструмент бизнес-презентаций, поскольку вместо шлема здесь нужен джойстик, мышь или шаровой манипулятор, при помощи которых пользователь может повернуть трехмерную модель на мониторе на все 360°. С помощью такой системы легко показать модель будущего здания или проект автомобиля (рис. 16).

Рис. 16. Настольная VR-система (Schneider Digital)

Визуально согласованный дисплей (visually coupled display) размещается прямо перед глазами пользователя и изменяет картинку согласно движениям головы. Дисплей снабжен стереофоническими наушниками и системой отслеживания направления взгляда и фокусирует изображение, на которое направлено внимание пользователя (рис. 17).

Рис. 17. Визуально согласованный дисплей (Vive)

На сегодняшний день такие виртуальные системы медленно получают массовое распространение за счет удешевления устройств, увеличения мощности видеокарт, улучшения качества моделируемого мира. Пока наиболее активно VR-системы используются инструкторами для обучения пилотированию сложной транспортной техники (самолетов, спецтехники, судов и др.), военными для имитации боевых событий и действий. Также VR-системы применяют в качестве тренажеров для быстрого обучения ведению боя в ситуациях, создаваемых имитациями.

Другая область, где виртуальная реальность уже нашла свою нишу, – индустрия развлечений. Здесь виртуальные миры становятся логическим продолжением традиционных компьютерных игр, особенно это касается

игр «от первого лица», где играющий получает возможность оказаться в центре событий и почти в буквальном смысле прочувствовать эти события.

Одна из основных проблем в освоении виртуальной реальности состоит в том, чтобы частично совпадающие (перекрывающиеся) данные, поступающие в мозг человека от различных рецепторов, были удовлетворительны в информационном отношении. Диссонанс восприятия, когда сигналы разноречивы, может вызвать дезориентацию, растерянность и даже болезнь. Сегодня эти дефекты еще довольно велики, потому что несмотря на все достижения технологий, в современной виртуальной реальности пока не очень много реального, и вы легко отличите виртуальный объект от настоящего. Кроме того, большая часть компьютеров пока обладает замедленной реакцией на входной сигнал, а VR-системы, работающие в реальном времени, то есть позволяющие имитировать взаимодействие с окружающей виртуальной средой в режиме и со скоростью, подобными реальным, пока очень дороги. Применение VR-систем также сдерживается низкой разрешающей способностью дисплеев на жидких кристаллах. Но это технологические трудности, которые в ближайшие годы будут постепенно преодолены.

Возможно, более сложная проблема с системами виртуальной реальности имеет психологическую природу и состоит в приучении пользователей к новым технологиям, внедряющимся в их работу. Предположим, корпорация располагает двумя макетами операционных технологий для какой-то производительной деятельности. Один трехмерный и усложненный; другой простой и пока двумерный. Проблема заключается в том, захотят ли люди носить очки и перчатки, чтобы в них заниматься повседневными делами. Опыт трехмерных фильмов позволяет ожидать сопротивление.

Надевать очки и перчатки соглашаются те, кому приходится переодеваться в силу профессиональной деятельности: пилоты и астронавты. Дети,

конечно, с удовольствием наденут костюмы и шлемы, делающие видеоигры более реалистичными. Однако проблема преодоления психологического барьера всегда сопровождает рывок технологий, и можно рассчитывать, что со временем она будет преодолена при условии, что пользователи виртуальных систем будут не просто потребителями, а творцами и создателями своих виртуальных миров [20, 36].

1.2. Сферы применения технологий виртуальной реальности

1.2.1. Применение ВР систем в промышленности и производстве

Одной из первых на эксперимент по применению виртуальной реальности на производстве решилась американская корпорация General Motors. Риск себя оправдал: созданный в 1994 г. в Детройте центр виртуальной реальности обошелся концерну в 5 млн долл., а экономия при разработке новых моделей машин составила около 80 млн.

Применение системы виртуальной реальности позволяет убрать из процесса разработки новой модели такие операции, как создание пластилинового макета, продувка модели в натуральную величину в аэродинамической трубе и краш-тесты. Все эти манипуляции инженеры и дизайнеры производят в виртуальном пространстве, где изменениям подвергается не физический, а электронный прототип автомобиля.

Сходным образом решаются и проблемы эргономики салона, компоновки моторного отсека и ремонтопригодности узлов и агрегатов будущей машины. Например, если какой-либо узел оказывается труднодоступным, модель от инженеров вновь поступает к дизайнерам, которые «на лету»

корректируют элемент кузова, мешающий подобраться к нужному месту. Затем электронная модель вновь передается инженерам.

Вслед за General Motors центрами виртуальной реальности обзавелись Volkswagen и Ford. Так, компания Ford признает, что внедрение системы виртуальной реальности в дизайнерских центрах в Меркенихе (Германия) и Дантоне (Великобритания) позволило сократить время разработки нового автомобиля с 42 до 24 месяцев. Самым впечатляющим результатом внедрения этой технологии стала Audi A3 (фирма Audi входит в состав группы Volkswagen), которая разрабатывалась почти без использования реальных моделей (рис. 18).

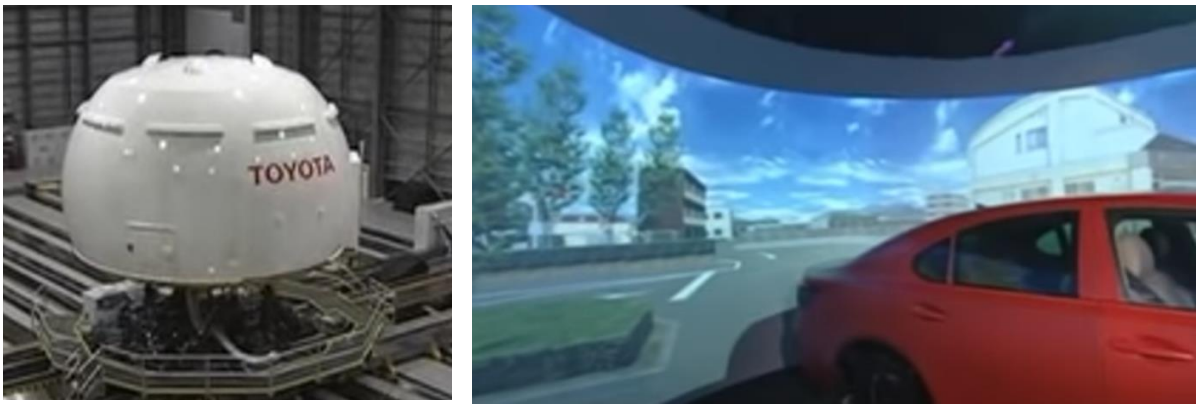
а)

б)

Рис. 18. Ручная модель автомобиля (а);
компьютерная модель автомобиля (б)

Компания Toyota в 2007 г. представила компьютерный симулятор, позволяющий испытать новые модели в действии, но при этом все происходит с применением VR-системы. Симулятор представляет собой купол высотой 4,5 м и диаметром 7 м, внутри которого помещается самый настоящий автомобиль. Вместе с реалистичной аудиосистемой устройство купола создаёт полностью реалистичную картину окружающего мира. Водитель, находясь в автомобиле внутри купола и управляя, на самом деле никуда не едет, однако испытывает все ощущения, как если бы двигался по настоящей дороге. Конечно, физическое взаимодействие, приводящее в реальности к

летальному исходу, в этой капсуле невозможно, однако ощутить дорогу, повороты, торможение, ускорение здесь можно с полным успехом (рис. 19).



а) б)

Рис. 19. Toyota´s Driving Simulator, снаружи (а), внутри (б)

Одно из интереснейших приложений виртуальной реальности – это моделирование краш-тестов современных автомобилей. В связи с развитием технологий и методов обработки компьютерной графики ставится задача ускорения процесса производства и возможности воспроизведения натурных экспериментов в компьютерной практике.

Плюсами, несомненно, является возможность проведения краш-тестов без построения специального полигона со всем необходимым оборудованием (авто, манекены, топливо, камеры для высокоскоростной съемки, осветительное оборудование и т.д.). Также нет необходимости в содержании большого штата сотрудников для поддержания всей этой инфраструктуры. Появляется возможность многократного проведения тестов при небольшом изменении параметров тестируемой модели как авто, так и систем безопасности, при натурном эксперименте возможно было только однократное тестирование.

Еще одним плюсом является возможность в реальном времени просматривать деформации и повреждения всех частей авто без длительного

разбора и демонтажа. Как итог получается снижение затрат на производство, повышение точности проектируемых деталей, учет всех возможных ситуаций посредством моделирования всех возможных проекций движения авто (а их может быть десятки тысяч) и анализа поведения и повреждений для пассажиров и водителя (рис. 20).

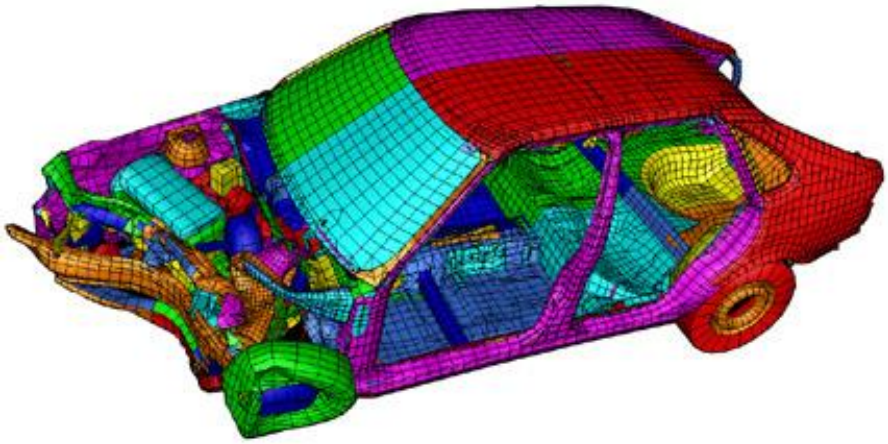


Рис. 20. Моделирование краш-теста в виртуальной реальности

Современные системы виртуальной реальности, используемые в производстве, – новый этап развития хорошо известных систем автоматизированного проектирования и моделирования. Все дорогостоящие приспособления (проекционные системы, специальные шлемы, перчатки, костюмы, благодаря которым передается не только изображение, но и звук и тактильные ощущения) являются обычными устройствами ввода/вывода информации. Однако системы виртуальной реальности имеют одно принципиальное отличие: ни одна установка автоматизированного проектирования и моделирования не позволяет человеку управлять поведением модели в реальном времени.

Самые дешевые персональные системы виртуальной реальности могут стоить от единиц до десятков тысяч долларов. Однако у них есть существенный недостаток: такие системы обычно не позволяют работать коллек-

тивно. Но без коллегиальности при принятии решений теряется возможность так организовать непрерывный производственный цикл, чтобы вся подготовительная работа происходила в виртуальном мире, а в реальный мир новое изделие попадало бы уже в виде мелкосерийных образцов.

Полнофункциональная система виртуальной реальности, точнее – центр виртуальной реальности (оборудование и программное обеспечение), стоит от нескольких десятков тысяч до нескольких миллионов долларов. Разработка виртуального мира в зависимости от его сложности и специфичности обойдется от 2–3 до 100 тыс. долл. Что производство получит взамен? Прежде всего сократятся сроки разработки. Скажем, применение систем виртуальной реальности в автомобилестроении позволяет сократить цикл подготовки новой модели к серийному производству с 18 месяцев (в США и Западной Европе) до полугода. При этом гораздо меньше времени требуется и на доводку автомобиля, поскольку все вопросы по эргономике салона и ремонтопригодности узлов и агрегатов решаются на этапе электронного прототипирования (доводка серийной машины при использовании стандартных технологий занимает от нескольких месяцев до года).

Когда разработки модели ускоряется, возрастает конкурентное преимущество компании.

Еще одна обширная сфера применения виртуальной реальности – это работа с геоинформационными данными. Исследование ландшафтов, поиск полезных ископаемых, анализ сейсмической активности и многое другое, проводятся с использованием VR-систем.

Компания Reality AS и Фраунгоферовский институт медиакоммуникаций разработали системы виртуальной реальности, которые позволяют создавать уникальную и мощную среду для интерактивного проектирования скважин, оперативного управления геологическими изысканиями и геофизического анализа.

Инструментарий Inside Reality от Reality AS обеспечивает необычный, увлекательный и интуитивный способ работы с геоинформационными данными: пользователи взаимодействуют с моделью месторождения, используя естественные движения руки и тела и имитируя ходьбу, указание и выбор объекта. Инструментарий Фраунгоферовского института медиакоммуникаций VR-Geo использует специальное устройство для управления геоинформационными данными – кубическую мышь, которая имеет двенадцать степеней свободы и позволяет легко и быстро перемещаться «внутри земной коры».

Нефтегазовая и металлургическая компания с мировым именем Norsk Hydro достигла впечатляющих результатов с помощью Inside Reality. Ей удалось значительно (в некоторых случаях на 90%) сократить время проектирования горизонтальных скважин и, как следствие, добиться более аккуратного планирования и существенного увеличения нефтедобычи.

Нефтяные и газовые компании находят инновационные способы применения технологии виртуальной реальности для подземных исследований, обучения и симуляций, а также для разработки и изобретения виртуальных производственных процессов и продуктов (рис. 21).

Рис. 21. VR в геологоразведке

Baker Hughes, a GE company (BHGE) использует VR-технологии для симулирования нефтяного и газового оборудования в учебных помещениях. VR помогает компании сосредоточиться на цифровом проектировании оборудования. Технология обеспечивает повышенный уровень точности и понятности работы с оборудованием ещё до этапа его монтажа.

Газпром использует VR-технологии в своих маркетинговых и дистрибьюторских операциях, что позволяет повысить вовлеченность клиентов. Чтобы обогатить пользовательский опыт и сблизить конечного пользователя и компанию, Газпром создал платформу с VR, отображающую, как топливо добирается до конечного потребителя. С помощью VR-очков потребители на заправочных станциях Газпрома могут в короткой виртуальной экскурсии изучить всю цепочку производства топлива (рис. 22).

Рис. 22. VR concept, Газпромнефть

В СИБУРе возможности виртуальной реальности используются в обучении сотрудников, обеспечивают:

- запоминание алгоритмов и последовательностей действий для новых сотрудников, для нового оборудования со сложной последовательностью действий;

• сохранение знаний редкого повторения, но критической важности (пуск или остановка компрессора);

• обучение новых сотрудников из удаленных локаций без разработки сложной инфраструктуры (рис. 23) [21].

Рис. 23. ВР-технологии в СИБУРе

1.2.2. VR-системы для обучения и тренировки

Отработка навыков военных, пилотов, спасателей, водителей большегрузного транспорта в настоящее время осуществляется и с применением систем виртуальной реальности. Этот процесс экономичен, безопасен, позволяет достичь максимальных результатов.

Пентагон вкладывает достаточно много средств в разработку систем виртуальной реальности как для отработки навыков ведения боя в различных условиях, так и для бесконтактного управления боем.

Например, управление беспилотным самолетом-разведчиком Predator компании Boeing осуществляется из дистанционного центра

управления боем. Центр фактически является системой виртуальной реальности и позволяет оператору вести военные действия, находясь за сотни километров (рис. 24).

Системы виртуальной реальности применяются и спецподразделениями для борьбы с терроризмом, отработки и моделирования операций.

а) б)

Рис. 24. Беспилотник Predator (а);
центр управления беспилотником Predator (б)

Например, система What if Scenario Visualization («Система визуализации сценарного моделирования операций») компании EON Reality позволяет в реальном времени моделировать, планировать и координировать выполнение операций группой специалистов по принципу «а что, если попробовать другой вариант». Эта система дает возможность моделировать и планировать нестандартные действия, отрабатывать взаимодействие специальных групп. Аналогичные комплексы существуют и для решения более узких задач: обеспечения безопасности ядерных объектов и противодействия химическим и биологическим атакам (рис. 25).

Рис. 25. VR-система «What if»

Система виртуальной реальности Accident Training in nuclear power plant facilities (Система тренировки устранения аварийных ситуаций на ядерных объектах) позволяет имитировать и моделировать различные внештатные ситуации на ядерных объектах. Позволяет поднять уровень безопасности функционирования ядерных объектов и снизить затраты на обучение и отработку координации действия (рис. 26).

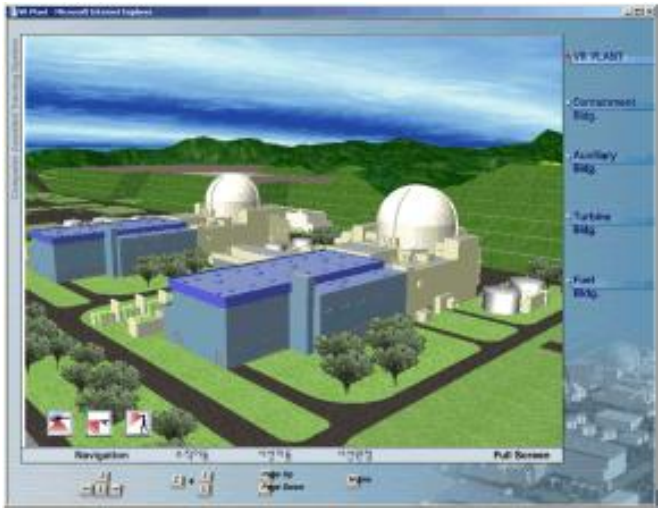


Рис. 26. Интерфейс Accident Training in nuclear power plant facilities

Chemical and Biological Terrorism Countermeasures (Система контрмер химическому и биологическому терроризму) – комплексная система моделирования, имитации и предотвращения биологических и химических атак. Позволяет повысить безопасность важных объектов и резко снизить стоимость обучения по отработке действий специалистов химической и биологической защиты.

В настоящее время полноценные тренажёры, используемые для отработки всего полётного процесса, представляют собой сложные технические комплексы, сочетающие широкоугольные экраны, подвижные платформы (с тремя, а иногда с шестью степенями свободы), контроллеры с обратной связью (например, штурвалы, на которые надо оказывать такое же силовое воздействие, как в реальном полёте) и приборные доски, с точностью имитирующие поведение настоящего оборудования на борту воздушного судна.

Некоторые системы, такие, например, как система NASA Vertical Motion System позволяют имитировать любые лётные средства (вплоть до космических кораблей), а также работу диспетчерских служб (модуль FutureFlight Central – это как раз тренажёр для диспетчеров).

В России тоже выпускаются авиационные тренажёры, причём, по утверждениям разработчиков, соответствующие всем международным стандартам. В частности, компания «Транзас» выпускает комплексные тренажёры класса FFS (Full Flight Simulation), а также процедурные симуляторы, используемые для отработки конкретных ситуаций (рис. 27). Особый интерес представляет система визуализации «Аврора», обеспечивающая фотореалистичное визуальное представление закабинного пространства [44].

Представление ландшафта основывается на картографических данных, данных аэрофотосъёмок конкретных регионов, также учитывается графическая база данных с множеством текстур и объектов, которая позволяет

моделировать разнообразные типы поверхности земли в различные времена года. Разработан модуль импорта объектов из 3ds Max, собственные средства моделирования ландшафта и разных аэропортов. Географический дизайн самостоятельно выбирает заказчик.

Рис. 27. Кабинный симулятор компании «Транзас»

Для создания большей реалистичности моделируются и все возможные погодные условия.

Использование даже самого дорогостоящего симулятора при обучении потом окупается на практике: пилоты имеют возможность отрабатывать любые штатные и нештатные ситуации, не рискуя ни своими жизнями, ни жизнями тех, кого они перевозят.

1.2.3. Применение VR-систем в образовании и исследовании

VR-системы нашли свое применение в научном секторе. Так, в психотерапии при изучении фобий и психических расстройств пациентов приме-

няется виртуальная среда, в которой создается ситуация, требующая психотерапевтической работы и контролируемая психотерапевтом. Например, при лечении фобий моделируется ситуация, в которой проявляется конкретная фобия: прогулки по крышам небоскребов (при боязни высоты), авиаперелет (при страхе перед полетами на самолете), «общение» с пауками (при арахнофобии) и т.п.

В традиционной когнитивно-бихевиоральной терапии терапевт устраивает терапевтические сессии, в которых создаются ситуации, вызывающие страхи клиента, начиная с наименее травматичных и продвигаясь по иерархии к ситуациям, вызывающим большую тревогу. Например, при лечении боязни высоты начинают с моделирования ситуации, в которой используется трехэтажный дом, а заканчивают использованием в ситуации крыши небоскреба. Психотерапевт в этот момент находится рядом с клиентом и при помощи беседы помогает ему прорабатывать свой страх, а при необходимости успокаивает его [20] (рис. 28).

Рис. 28. Лечение арахнофобии в VR

С помощью виртуальных систем врачи учатся и исследуют пациентов, создают 3D-узи, анализируют томографию. Проведение сложных операций, требующих тренировки, невозможно без таких хирургических систем как модель MIST VR фирмы MUSE Technologies, CAVEman.

В 1989 г. методами виртуальной реальности была проведена первая лапароскопическая операция на желчном пузыре.

Исследование взаимодействия медицинских препаратов также проводятся с помощью компьютерных и виртуальных систем.

Весьма перспективно применение тактильных устройств в VR-системах обучения зубных врачей. Обучающую стоматологическую VR-систему (Virtual Reality Dental Training System – VRDTS) намерены создать фирма Teneo Computing, специализирующаяся в области трехмерной VR, и стоматологическая школа при Гарвардском университете – признанный в мире эксперт в области зубоврачебной практики, исследований и обучения.

Teneo Computing впервые объединила трехмерные тактильные интерфейсы со стереоизображением и получила реальные трехмерные модели взаимодействия. Используя вместо мыши рабочий инструмент тактильного прибора (это может быть сверло или четыре других зубоврачебных инструмента), будущий стоматолог может не только найти виртуальный объект, но и «почувствовать» его. Система способна моделировать зубы с различными свойствами эмали, мягкой пульпой и даже с повреждениями. С помощью такой системы студенты смогут неоднократно сверлить один и тот же зуб и даже, увеличив изображение, увидеть и оценить результаты своей работы (рис. 29).

Рис. 29. VR в стоматологии

Возможно, в новой системе обучения трехмерное тактильное устройство 3D TouchTM будет работать с действующими рабочими станциями компании Sun Microsystems.

Varwin Education – простой в освоении конструктор для создания ВР-приложений и управления ими, развивающий навыки программирования с помощью визуальной среды Blockly. Российское ПО Varwin Education создано на базе Varwin XRMS – системы управления 3D и ВР-контентом. VARWIN XRMS – платформа для создания и изменения проектов в виртуальной реальности, которая не требует от пользователя каких-либо специфических знаний. Платформа использует систему Unity 3D и редактор кода Blockly.

Varwin Education – это позволяет освоить программирование виртуальной реальности для детей 5–11 классов и для студентов. По мнению разработчиков, программирование в системе Varwin похоже на игру в Minecraft или Roblox. Большая информационная поддержка педагогов, школьников, студентов позволяет легко по вебинарам или самостоятельно освоить основные операции по работе с системой. Разработанные ВР-приложения воспроизводятся как в шлемах, так и на десктопе. Varwin Education входит в шорт-лист программного обеспечения Московского детского чемпионата KidSkills по технологиям разработки виртуальной и дополненной реальности (рис. 30).

Рис. 30. Разработка проекта в Varwin

Возможность проведения имитационного моделирования позволяет исследовать области, недоступные пока человечеству: пространство и законы вселенной, происхождение и формирование нашей планеты и многое другое.

Виртуальные лаборатории, как отдельное направление в разработке программного обеспечения, применяется при обучении школьным дисциплинам. Виртуальные лаборатории по физике, химии, биологии, информатике, математике, геометрии применяются как тренажеры в школах, вузах [24, 32]. Виртуальные лаборатории позволяют увеличить степень самостоятельности обучающихся, увеличить их вовлеченность в процесс решения задач с практическим содержанием [33].

Как видно из обзора, технологии виртуальной реальности постепенно занимают определенную нишу во многих отраслях производства, медицины, исследований и обучения. Можно прогнозировать, что направление технического развития VR-систем будет идти в сторону совершенствования возможностей графического отображения и одновременно постепенного удешевления систем. Будут совершенствоваться устройства воздействия на органы чувств человека. Все большая виртуализация и повсеместное внедрение Интернета приведет к заметным переменам в социальной структуре общества: исчезнут различия в развитии между государствами, имеющими доступ в Интернет, произойдет стирание границ между людьми, упразднение государственных и прочих иерархических структур.

1.3. Особенности технологии разработки систем виртуальной реальности

1.3.1. Составляющие систем виртуальной реальности

Полноценная система виртуальной реальности обычно базируется на следующих составляющих:

- графика;
- управление;
- искусственный интеллект (логика, физика);
- звук.

Исключение хотя бы одной из данных компонент делает процесс работы с VR-системой весьма ограниченным, а иногда и попросту бессмысленным. Трудно представить себе VR-систему для тренировки пилотов без высокого уровня моделирования физических явлений или работу хирурга в условиях плохого качества графики. Поэтому стремление разработчиков совершенствовать вышеперечисленные компоненты оправдано.

Разработкой VR-систем занимаются крупные инженерные и софтверные компании, использующие результаты компьютерных наук в области графических алгоритмов, автоматизированного управления, компьютерного моделирования, искусственного интеллекта, дискретизации звуковой информации. Разработкой систем виртуальной реальности занимаются специалисты из областей знаний, не связанных с техникой: психологи, дизайнеры, эксперты предметной области и многие другие.

Вот, например, типичный состав коллектива разработчиков VR-систем (рис. 31):

- руководитель проекта;
- сценарист;
- художники-дизайнеры;
- моделлеры;

- аниматоры;
- программисты;
- звукорежиссер [17].

Рис. 31. Разработка VR-систем

И это только специалисты-разработчики. Если учитывать еще экспертов предметной области (пилотов, врачей и др. специалистов, для которых предназначена будущая система), тестеров, фотографов и пр., то получится внушительный список.

Далее мы рассмотрим особенности каждой из компонент систем виртуальной реальности.

1.3.2. Графика для виртуальной реальности

Представление реалистичной 3D-картинки в VR-системах возможно с помощью двух технологий: стереоизображений и генерируемой 3D-графики.

Стереоизображение – картина или видеоряд, использующее два отдельных изображения, позволяющих достичь стереоэффекта. Чтобы создать стереоизображение в программе трёхмерного моделирования, надо сделать двойной рендеринг сцены – с двух камер, соответствующих глазам наблюдателя (рис. 32).

Рис. 32. Пример стереоизображения

С помощью каких механизмов мы воспринимаем мир объёмным? Перечислим факторы, дающие нам информацию об объёме.

- Геометрическая перспектива и знание реальных размеров наблюдаемых объектов.
- Воздушная перспектива.
- Тени и блики.
- Движение наблюдателя и движение предмета наблюдения.
- Аккомодация хрусталика (приспособление органа либо организма в целом к изменению внешних условий).
- Бинокулярность зрения – наличие не одного, а двух глаз. Мозг сопоставляет изображения, которые видит правый и левый глаз (рис. 33).

В обычной фотографии для передачи используются первые три фактора. В кино для передачи ощущения пространства очень важен четвёртый фактор, для его реализации операторы плавно перемещают камеры. Главным фактором стереоизображений является последний из перечисленных.

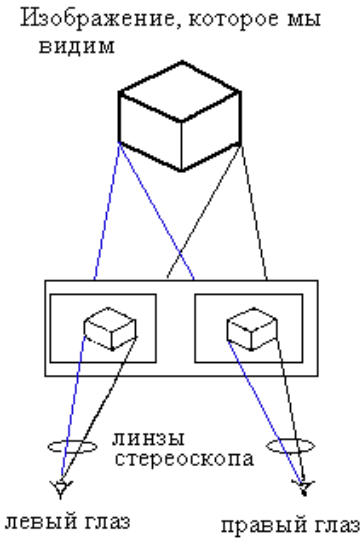


Рис. 33. Формирование изображения при бинокулярности

Создавать стереоизображения можно двумя способами: с помощью одной камеры (использование специальной стереокамеры, затворной или зеркальной стереонасадок) или с помощью двух камер, расположенных на небольшом расстоянии друг от друга (рис. 34).

Рис. 34. Стереокамера

Сложности возникают тогда, когда нужно увидеть снятое таким образом стереоизображение. Для этого необходимо, чтобы каждый глаз видел предназначенное для него изображение и не видел изображение

для другого глаза. Без специальной тренировки глаза у человека смотрят, как правило, так, как им предписано природой, нисколько не соизмеряясь с нашими намерениями. И вместо объемного изображения видят два плоских [23].

Просмотр стереоизображений осуществляется обычно с помощью специальных устройств. Это анаглифные очки, поляризационные очки, затворные очки, виртуальный шлем, 3D мониторы, 3D экраны (рис. 35).

а) б) в)

Рис. 35. Анаглифные очки (а); поляризационные очки (б); VR шлем (в)

Помимо стереоизображений 3D виртуальную реальность можно создать с помощью редакторов трехмерной компьютерной графики.

Трехмерная компьютерная графика представляет собой сочетание растровой и векторной компьютерной графики с алгоритмами для быстрой перерисовки основного графического профиля и внешнего вида, позволяющими оперативно изменять перспективу и точку наблюдения, такой процесс назван трехмерной визуализацией.

Чтобы создать иллюзию трехмерного пространства, объекты на экране компьютера строят на основе «проволочного» каркаса, составленного из масштабируемых линий или многоугольников, создаваемых с привлечением средств векторной графики (рис. 36). Для придания желаемого внешнего вида «проволочный» каркас закрывается поверхностным слоем [13].

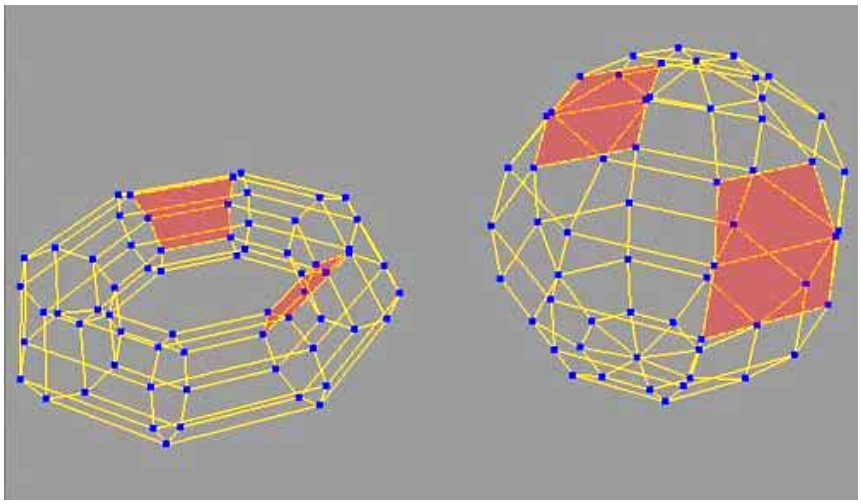


Рис. 36. Полигональные модели

Основными этапами создания 3D модели являются следующие:

- моделирование;
- анимация (изменение положения объектов, которые присутствуют в трехмерном пространстве, во времени);
- съемка (просмотр сцены с разных точек);
- освещение (расположение источника света, сила источника, отражение);
- текстурирование;
- визуализация (рендеринг).

Алгоритмы 3D-графики развиваются и улучшаются постоянно. Рассмотрим наиболее популярные алгоритмы.

Управление прозрачностью (альфа канал) – это формирование цвета пиксела с учетом величины прозрачности объекта по следующей формуле [22]:

R = B*(1-T)+F*T, где

B – исходный цвет пиксела,

F – цвет накладываемого пиксела,

T – прозрачность накладываемого пиксела от 0 до 1 (рис. 37).

Рис. 37. Управление альфа-каналом

Глубина изображения (z-буфер). В 1974 г. Эд Кэтмул предложил алгоритм Z-буферизации – в компьютерной трёхмерной графике способ учёта удалённости элемента изображения. Представляет собой один из вариантов решения «проблемы видимости». Очень эффективен и практически не имеет недостатков, если реализуется аппаратно [13].

Все алгоритмы такого рода так или иначе включают в себя сортировку, причем главная сортировка ведется по геометрическому расстоянию от тела, поверхности, ребра или точки до точки наблюдения или картинной плоскости. Идея, положенная в основу сортировки по расстоянию, заключается в том, что чем дальше расположен объект от точки наблюдения, тем больше вероятность, что он будет полностью или частично заслонен одним из объектов, более близких к точке наблюдения. После определения расстояний или приоритетов по глубине остается провести сортировку по горизонтали и по вертикали, чтобы выяснить, будет ли рассматриваемый объект действительно заслонен объектом, расположенным ближе к точке наблюдения. Эффективность любого алгоритма удаления в значительной мере зависит от эффективности процесса сортировки (рис. 38).

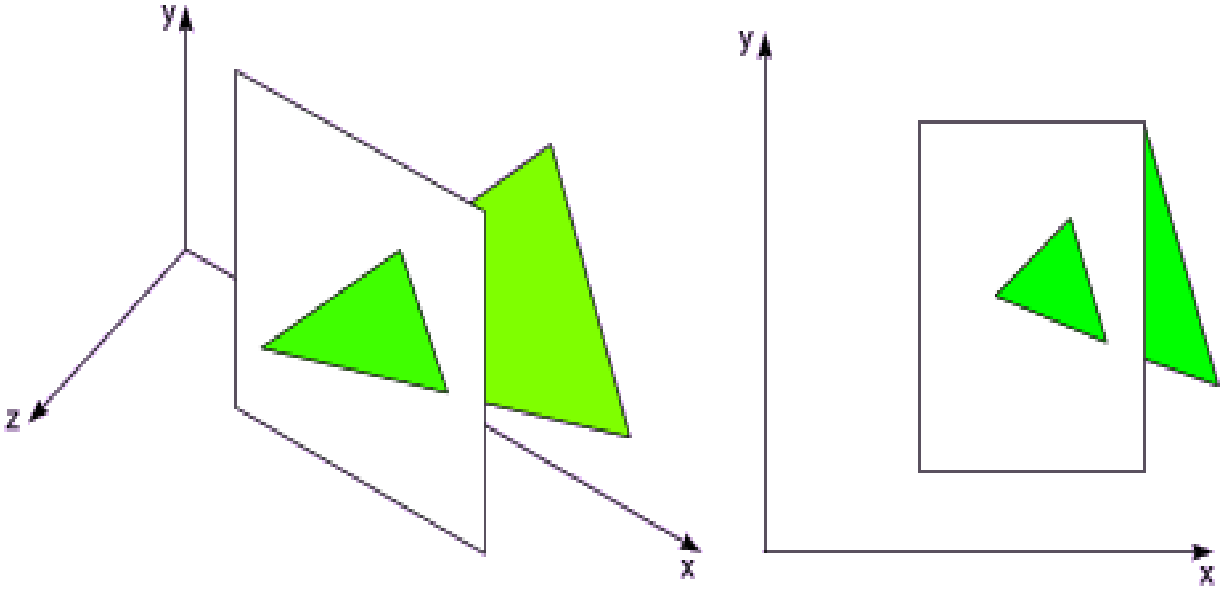


Рис. 38. Расчет удаленности объекта от точки наблюдения

Z-буфер представляет собой двумерный массив (матрицу), каждый элемент которого соответствует пикселу на экране. Когда видеокарта рисует пиксел, его удалённость просчитывается и записывается в ячейку Z-буфера.

Если пикселы двух рисуемых объектов перекрываются, то их значения глубины сравниваются и изображается тот, который ближе, а его значение удалённости сохраняется в буфер. Получаемое при этом графическое изображение носит название z-depth map. Каждый пиксел этого изображения может принимать до 256 значений серого. По ним определяется удалённость от зрителя того или иного объекта трехмерной сцены.

Карта широко применяется в постобработке для придания объёмности и реалистичности и создаёт такие эффекты, как глубина резкости, атмосферная дымка и т.д. Также карта используется в 3D-пакетах для текстурирования, делая поверхность рельефной (рис. 39).

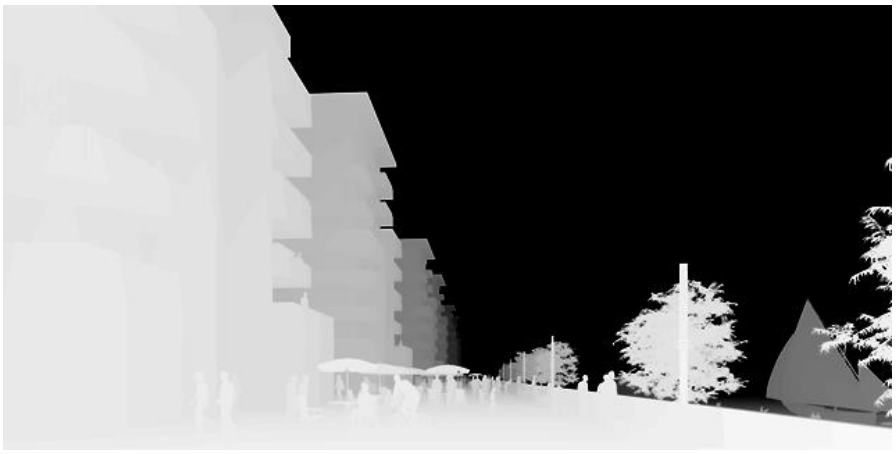


Рис. 39. Матрица z-буфера изображения

Расчет освещенности поверхности

Если предположить, что источник света находится на бесконечности, то лучи света, падающие на поверхность, параллельны между собой. Если к этому добавить условие, что наблюдатель находится в бесконечно удаленной точке, то эффектом ослабления света с увеличением расстояния от источника также можно пренебречь. Кроме того, такое положение наблюдателя означает еще и то, что векторы, направленные от разных точек поверхности к наблюдателю, также будут параллельны. При выполнении всех этих условий, плоская грань во всех точках имеет одинаковую интенсивность освещения, поэтому она закрашивается одним цветом. Такое закрашивание называется **плоским** [8].

Если мы аппроксимируем некоторую гладкую поверхность многогранником, то при плоском закрашивании неизбежно проявятся ребра, поскольку соседние грани с различными направлениями нормалей имеют разный цвет. Эффект полос Маха дополнительно усиливает этот недостаток. Для его устранения при использовании этого способа закрашивания можно лишь увеличить число граней многогранника, что приводит к увеличению вычислительной сложности алгоритма (рис. 40).

а) б)

Рис. 40. Плоская модель затенения: а) 2000, б) 32000 треугольников

Один из способов устранения дискретности интенсивностей закрашивания был предложен **Гуро**. Его метод заключается в том, что используются не нормали к плоским граням, а нормали к аппроксимируемой поверхности, построенные в вершинах многогранника. После этого вычисляются интенсивности в вершинах, а затем во всех внутренних точках многоугольника выполняется билинейная интерполяция интенсивности.

Метод сочетается с алгоритмом построчного сканирования. После того как грань отображена на плоскость изображения, для каждой сканирующей строки определяются ее точки пересечения с ребрами. В этих точках интенсивность вычисляется с помощью линейной интерполяции интенсивностей в вершинах ребра. Затем для всех внутренних точек многоугольника, лежащих на сканирующей строке, также вычисляется интенсивность методом линейной интерполяции двух полученных значений. На рисунке 41 показан плоский многоугольник с вычисленными значениями интенсивностей в вершинах.

а) б)

Рис. 41. Метод Гуро: а) 2000, б) 32000 треугольников

К недостаткам метода Гуро следует отнести то, что он хорошо работает только с диффузной моделью отражения. Форма бликов на поверхности и их расположение не могут быть адекватно воспроизведены при интерполяции на многоугольниках. Кроме того, есть проблема построения нормалей к поверхности. В алгоритме Гуро нормаль в вершине многогранника вычисляется путем усреднения нормалей к граням, примыкающим к этой вершине. Такое построение сильно зависит от характера разбиения.

Фонг предложил вместо интерполяции интенсивностей произвести интерполяцию вектора нормали к поверхности на сканирующей строке. Этот метод требует больших вычислительных затрат, поскольку формулы интерполяции применяются к трем компонентам вектора нормали, но зато дает лучшую аппроксимацию кривизны поверхности. Поэтому зеркальные свойства поверхности воспроизводятся гораздо лучше.

Нормали к поверхности в вершинах многогранника вычисляются так же, как и в методе Гуро. А затем выполняется билинейная интерполяция в сочетании с построчным сканированием. После построения вектора нормали в очередной точке вычисляется интенсивность (рис. 42) [8].

а) б)

Рис. 42. Метод Фонга: а) 2000, б) 32000 треугольников

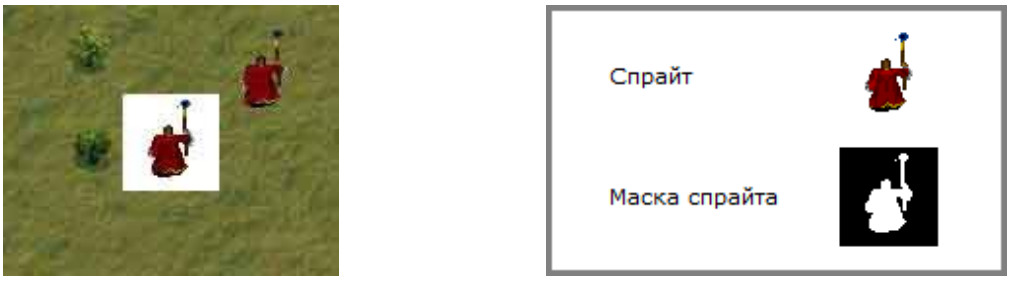
Текстурирование

Текстура – растровое изображение, накладываемое на поверхность полигональной модели для придания ей цвета, окраски или иллюзии рельефа. Приблизительно использование текстур можно представить как рисунок на поверхности скульптурного изображения. Использование текстур позволяет воспроизвести малые объекты поверхности, создание которых полигонами оказалось бы чрезмерно ресурсоёмким. Например, шрамы на коже, складки на одежде, мелкие камни, предметы на поверхности стен и почвы (рис. 43) [9, 42].

Рис. 43. Наложение текстуры на модель персонажа

Спрайты

Слово «спрайт» было придумано в 1970-е г. одним сотрудником компании Texas Instruments: новая микросхема TMS9918 могла аппаратно отображать небольшие картинки поверх неподвижного фона. Спрайт – чаще всего растровое изображение, свободно перемещающееся по экрану. Наблюдение спрайта под несоответствующим углом приводит к разрушению иллюзии. Легче всего воспринимать спрайт как перемещающуюся в пространстве проекцию какого-то объёмного тела так, что разница незаметна (рис. 44) [42].



а) б)

Рис. 44. Пример спрайта

1.4. Графические компоненты систем виртуальной реальности

Для создания 3D-графики в системах виртуальной реальности используются различные библиотеки. Наиболее популярными являются OpenGL и DirectX. Рассмотрим их особенности.

OpenGL является одним из самых популярных прикладных программных интерфейсов (API – Application Programming Interface) для разработки приложений в области двумерной и трехмерной графики.

Стандарт OpenGL (Open Graphics Library – открытая графическая библиотека) был разработан и утвержден в 1992 г. ведущими фирмами в области разработки программного обеспечения как эффективный аппаратно-независимый интерфейс, пригодный для реализации на различных платформах. Основой стандарта стала библиотека IRIS GL, разработанная фирмой Silicon Graphics Inc.

Библиотека насчитывает около 120 различных команд, которые программист использует для задания объектов и операций, необходимых для написания интерактивных графических приложений.

На сегодняшний день графическая система OpenGL поддерживается большинством производителей аппаратных и программных платформ. Эта

система доступна тем, кто работает в среде Windows, пользователям компьютеров Apple. Свободно распространяемые коды этой системы можно компилировать в большинстве операционных систем, в том числе в Linux.

Характерными особенностями OpenGL, которые обеспечили распространение и развитие этого графического стандарта, являются:

- стабильность (Дополнения и изменения в стандарте реализуются таким образом, чтобы сохранить совместимость с разработанным ранее программным обеспечением);
- надежность и переносимость (Приложения, использующие OpenGL, гарантируют одинаковый визуальный результат вне зависимости от типа используемой операционной системы и организации отображения информации. Кроме того, эти приложения могут выполняться как на персональных компьютерах, так и на рабочих станциях и суперкомпьютерах);
- легкость применения (Стандарт OpenGL имеет продуманную структуру и интуитивно понятный интерфейс, что позволяет с меньшими затратами создавать эффективные приложения, содержащие меньше строк кода, чем с использованием других графических библиотек. Необходимые функции для обеспечения совместимости с различным оборудованием реализованы на уровне библиотеки и значительно упрощают разработку приложений);
- наличие хорошего базового пакета для работы с трехмерными приложениями (Эта особенность упрощает понимание студентами ключевых тем курса компьютерной графики: моделирование трехмерных объектов, закрашивание, текстурирование, анимация и т.д. Широкие функциональные возможности OpenGL служат хорошим фундаментом для изложения теоретических и практических аспектов предмета) [48].

OpenGL ориентируется на две задачи: скрыть сложности адаптации различных 3D-ускорителей, предоставляя разработчику единый API; скрыть

различия в возможностях аппаратных платформ, требуя реализации недостающей функциональности с помощью программной эмуляции.

Функции OpenGL реализованы в модели «клиент – сервер». Приложение выступает в роли клиента, оно вырабатывает команды; а сервер OpenGL интерпретирует и выполняет команды. Сам сервер может находиться как на том же компьютере, на котором находится клиент (например, в виде динамически загружаемой библиотеки DLL), так и на другом (при этом может быть использован специальный протокол передачи данных между машинами).

GL обрабатывает и рисует в буфере кадра графические примитивы с учетом некоторого числа выбранных режимов. Каждый примитив – это точка, отрезок, многоугольник и т.д. Каждый режим может быть изменен независимо от других. Определение примитивов, выбор режимов и другие операции описываются с помощью команд в форме вызовов функций прикладной библиотеки [50].

Примитивы определяются набором из одной или более вершин (vertex). Вершина определяет точку, конец отрезка или угол многоугольника. С каждой вершиной ассоциируются некоторые данные (координаты, цвет, нормаль, текстурные координаты и т.д.), называемые атрибутами. В подавляющем большинстве случаев каждая вершина обрабатывается независимо от других.

Некоторые примитивы OpenGL

GL_POINTS – каждая вершина задает точку.

GL_LINES – каждая отдельная пара вершин задает линию.

GL_LINE_STRIP – каждая пара вершин задает линию (т.е. конец предыдущей линии является началом следующей).

GL_LINE_LOOP – аналогично предыдущему за исключением того, что последняя вершина соединяется с первой и получается замкнутая фигура.

GL_TRIANGLES – каждая отдельная тройка вершин задает треугольник.

GL_TRIANGLE_STRIP – каждая следующая вершина задает треугольник вместе с двумя предыдущими (получается лента из треугольников).

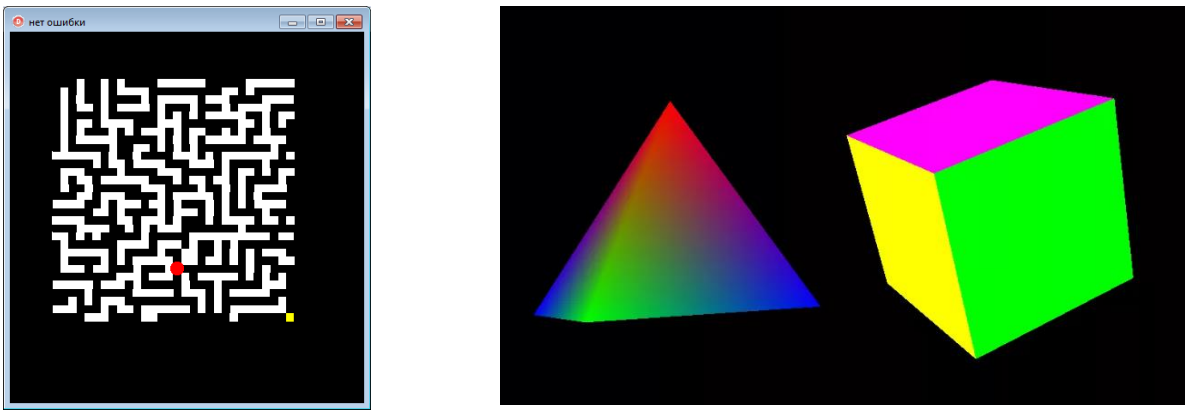
GL_TRIANGLE_FAN – каждый треугольник задается первой вершиной и последующими парами (т.е. треугольники строятся вокруг первой вершины, образуя нечто похожее на диафрагму).

GL_QUADS – каждые четыре вершины образуют четырехугольник.

GL_QUAD_STRIP – каждая следующая пара вершин образует четырехугольник вместе с парой предыдущих.

GL_POLYGON – задает многоугольник с количеством углов равным количеству заданных вершин.

Примеры рендеринга 3D-графики, сгенерированной с использованием OpenGL, представлены на рис. 45.



а) б)

Рис. 45. Пример программы «Лабиринт» (а); 3D-объекты (б)

В противовес бесплатной библиотеке OpenGL компания Microsoft решилась на создание собственной графической библиотеки. История появления DirectX берет свое начало в первой половине 1990-х г., когда компания Microsoft занималась разработкой операционной системы Windows 95, которая должна была прийти на смену MS-DOS.

Главным преимуществом MS-DOS было то, что она пользовалась популярностью у разработчиков игр. По мнению трех программистов Microsoft –

Крэйга Эйслера (Craig Eisler), Алекса Сен-Джона (Alex St. John) и Эрика Энгстрома (Erik Engstrom) – даже после выхода Windows 95 многие разработчики могли отдать предпочтение MS-DOS как более подходящей для создания игр платформе.

Чем же MS-DOS так нравился программистам? Все дело в том, что, программируя под MS-DOS, разработчики обращались напрямую к ресурсам компьютера, то есть имели прямой доступ к видеокарте, клавиатуре, мыши, звуковым устройствам и другим частям системы. Подобный подход использовался в программировании под консоли. Однако создание игр для компьютера осложнялось тем, что, в отличие от игр для приставок, здесь не было фиксированной конфигурации системы. Это приходилось учитывать при написании кода, что значительно усложняло жизнь девелоперам.

Первая итерация DirectX была очень упрощена относительно своих будущих версий. Она поддерживала вывод двумерной графики, звуков, а также обрабатывала данные, поступающие с различных манипуляторов.

Разработчики игр встретили DirectX довольно прохладно. Во-первых, они не были уверены, что Microsoft будет поддерживать API на протяжении долгого времени. Недоверие к Microsoft возросло после того, как компания свернула поддержку API WinG, который рассматривался как один из «помощников» в портировании игр с MS-DOS на Windows. Во-вторых, «девяносто пятая» была требовательней к аппаратной части, из-за чего производительность в играх, как правило, снижалась в сравнении с MS-DOS. Ну и, в-третьих, у MS-DOS было огромное количество сторонников, которые ни в какую не хотели программировать под Windows.

Долгое время версии DirectX не могли потеснить OpenGL. 2001 г. стал, пожалуй, переломным в истории развития DirectX. На протяжении предыдущих шести лет Microsoft находилась в роли догоняющего. И дело

было даже не в конкуренции с OpenGL, за которым развитие DirectX успевало, а в седьмой итерации в чем-то даже и превзошло OpenGL. DirectX догонял всю индустрию. Microsoft создавала каждую версию API с учетом архитектуры выпускаемых видеокарт. В таком положении трудно было сделать по-настоящему большой шаг вперед и вырваться в лидеры. И это не устраивало компанию. Поэтому Microsoft наладила сотрудничество с NVIDIA, и DirectX 8.0 появился почти одновременно с видеокартами GeForce 3 [1, 4].

Как итог программисты наконец-то увидели перспективу в DirectX. Многие посчитали, что API от Microsoft на самом деле может стать будущим разработки игр. Все потому, что программировать с помощью DirectX можно было с помощью обычного компьютера, в то время как для девелопмента под OpenGL необходима была рабочая станция. Вслед за NVIDIA свое внимание на DirectX обратила и ATI Technologies.

В целом, DirectX подразделяется на (рис. 46):

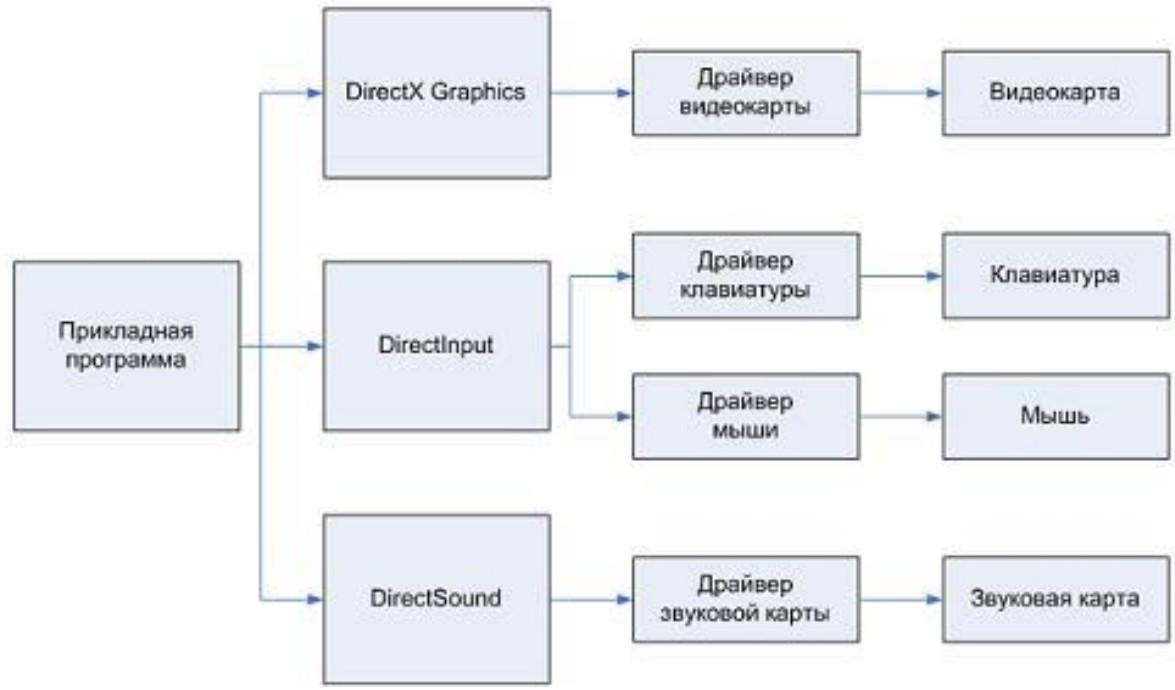


Рис. 46. Структура DirectX

- DirectX Graphics (набор интерфейсов, ранее (до версии 8.0) делившихся на (рис. 40): DirectDraw – интерфейс вывода растровой графики (его разработка давно прекращена); Direct3D (D3D) – интерфейс вывода трёхмерных примитивов);
- DirectInput (интерфейс, используемый для обработки данных, поступающих с клавиатуры, мыши, джойстика и прочих игровых контроллеров);
- DirectPlay (интерфейс сетевой коммуникации игр);
- DirectSound (интерфейс низкоуровневой работы со звуком формата Wave);
- DirectMusic (интерфейс воспроизведения музыки в форматах Microsoft).

DirectX11 появилась в 2009 г. и имел ряд улучшений:

- Tessellator (модуль тесселяции);
- Shader Model 5.0 (шейдерная модель версии 5.0);
- Compute Shader (вычислительные шейдеры);
- Multithreaded rendering (многопоточная визуализация);
- BC6 & BC7 (два новых формата сжатия текстур);
- Conservative oDepth (запись шейдера в буфер глубины) [4].

Тесселяция – увеличение количества полигонов. В основе тесселяции лежит идея о том, что объект, расположенный далеко от точки обозрения, будет менее детализирован потому что его тяжело рассмотреть. Но по мере его приближения количество треугольников в изображении объекта экспоненциально увеличивается, что улучшает его детализацию и делает объект более реалистичным.

Преимуществом этого метода является то, что, при рассмотрении просчитанного изображения, среднее число обработанных треугольников остается близким к устойчивому значению, так что игроку существенно реже приходится встречаться с резкими падениями производительности его системы (рис. 47, 48).

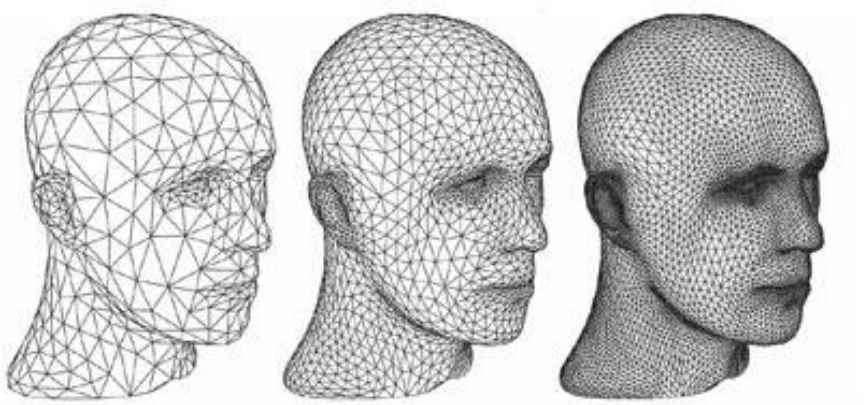


Рис. 47. Полигональная модель при тесселяции

а) б)

Рис. 48. Сцена без тесселяции (а); сцена с тесселяцией (б)

Шейдерная модель 5.0 (Shader Model 5.0) – это язык низкоуровневого программирования -, который использует объектно-ориентированные концепции, с тем чтобы облегчить разработку шейдеров и внести некоторые вспомогательные возможности.

Шейдер – это программа для одной из ступеней графического конвейера, используемая в трёхмерной графике для определения окончательных параметров объекта или изображения. Она может включать в себя произвольной сложности описание поглощения и рассеяния света, наложения текстуры, отражение и преломление, затенение, смещение поверхности и эффекты пост-обработки.

DirectX 12 вышел в 2015 г. и на сегодняшний день является последней версией библиотеки. В основе графической библиотеки лежат функции рисования, которые запускают графический конвейер – программно-аппаратное средство визуализации трехмерной графики. Аппаратная составляющая представлена видеоадаптером, программная – драйвером.

Графический конвейер можно представить в виде черного ящика, разделенного на этапы и выполняющего необходимые преобразования. Содержимое этого черного ящика может быть различным. Выполняемые преобразования зависят от назначения графической системы, стоимости, требуемого уровня универсальности и многих других факторов. Также, конкретный видеоадаптер – сложный механизм, правила работы которого зачастую известны лишь непосредственно производителю.

На сегодняшний день процесс визуализации трехмерной сцены выглядит в общих чертах следующим образом.

1. Преобразование вершин

Каждая вершина имеет определенный набор атрибутов, таких как позиция, цвет, текстурные координаты, вектор нормали или все векторы из касательного пространства и, возможно, некоторые другие. Трансформация вершин – это первая стадия графического конвейера. На этом этапе входными данными являются атрибуты конкретной вершины, над которыми производятся математические преобразования.

Эти операции включают трансформацию позиции вершины, генерацию и преобразование текстурных координат, расчет освещения для каждой отдельной вершины, а также любые другие операции, которые необходимо выполнить на уровне вершин. Каждая вершина обрабатывается параллельно с другими вершинами на доступных ядрах графического ускорителя. Основной результат вершинной программы – преобразовать координаты из модельного пространства в специальное пространство отсечения (clip space).

2. Построение примитивов и растеризация

Входные данные этого этапа – трансформированные вершины, а также информация об их соединении. Из этих данных осуществляется сборка геометрических примитивов. В результате получается последовательность треугольников, линий или точек. Над этими примитивами может производиться отсечение плоскостями, определенными в программе.

Также на этом этапе могут быть отброшены задние треугольники объектов. Определяются эти треугольники по направлению обхода вершин (по часовой стрелке или против). Направление обхода, соответствующее заднему треугольнику, задается через графическое API. Полигоны, прошедшие отсечение, могут растеризироваться.

3. Текстурирование и окрашивание

Над атрибутами примитивов, растеризированных в набор фрагментов, на этой стадии проводится необходимая интерполяция, а также последовательность математических преобразований и операций текстурирования, что определяет конечный цвет каждого фрагмента. Также на этом этапе может определяться новая глубина или даже исключение фрагмента из буфера кадра.

4. Пофрагментные операции

На этом этапе проводится ряд пофрагментных тестов, таких как тест отсечения (scissor test), тест трафарета (stencil test) и тест глубины (depth test). Эти тесты определяют конечный вид, цвет и глубину фрагмента перед обновлением экранного буфера. Если какой-либо тест проходит с ошибкой, то фрагмент не обновляется. После тестов выполняется операция смешивания, которая комбинирует финальный цвет фрагмента с текущим цветом пиксела, а итоговый результат записывается в экранный буфер. Операция смешивания выполняется на этом этапе, поскольку стадия текстурирования и окрашивания не имеют доступа к экранному буферу [1].

Среды для разработки VR – Unity 3D, Unreal Engine.

Unity 3D – это инструмент для разработки двумерных и трехмерных приложений, работающих под ОС Windows, Linux, OS X и др. (поддерживается более 20 операционных систем).

Редактор имеет простой интерфейс Drag-n-Drop (отладка приложения происходит прямо в редакторе) (рис. 49).

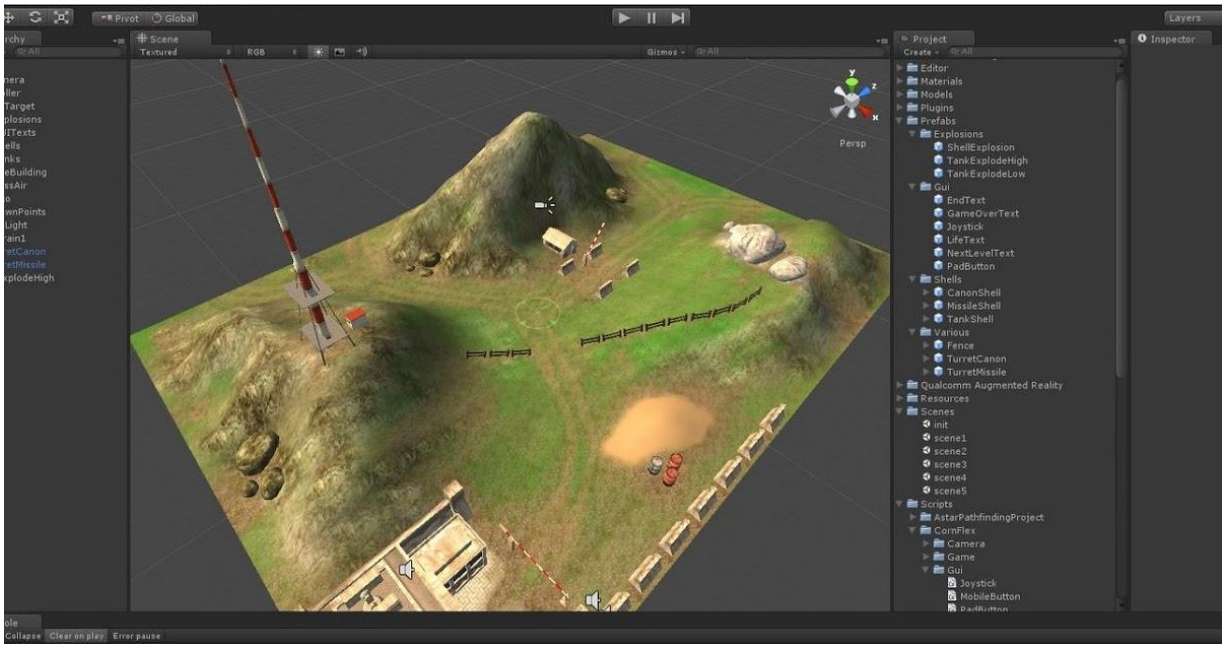


Рис. 49. Интерфейс Unity 3D

Поддерживаются два языка сценариев: JavaScript и C#. Физические расчеты производятся с использованием физической библиотеки PhysX от NVIDIA. Проект в Unity 3D делится на сцены (уровни) – отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев и настроек. Сцены могут содержать в себе как наполненные объекты (модели), так и пустые игровые объекты – объекты, которые не имеют модели («пустышки»). Объекты-модели, в свою очередь, содержат наборы компонентов, с которыми и взаимодействуют скрипты. Также у объектов есть названия (в Unity допускается наличие двух и более объектов с одинаковыми названиями), может быть тег (метка) и слой, на котором объект должен отображаться.

Так, у любого объекта на сцене обязательно присутствует компонент Transform. Он хранит в себе координаты местоположения, поворота и размеров объекта по всем трём осям. У объектов с видимой геометрией также по умолчанию присутствует компонент Mesh Renderer, делающий модель объекта видимой.

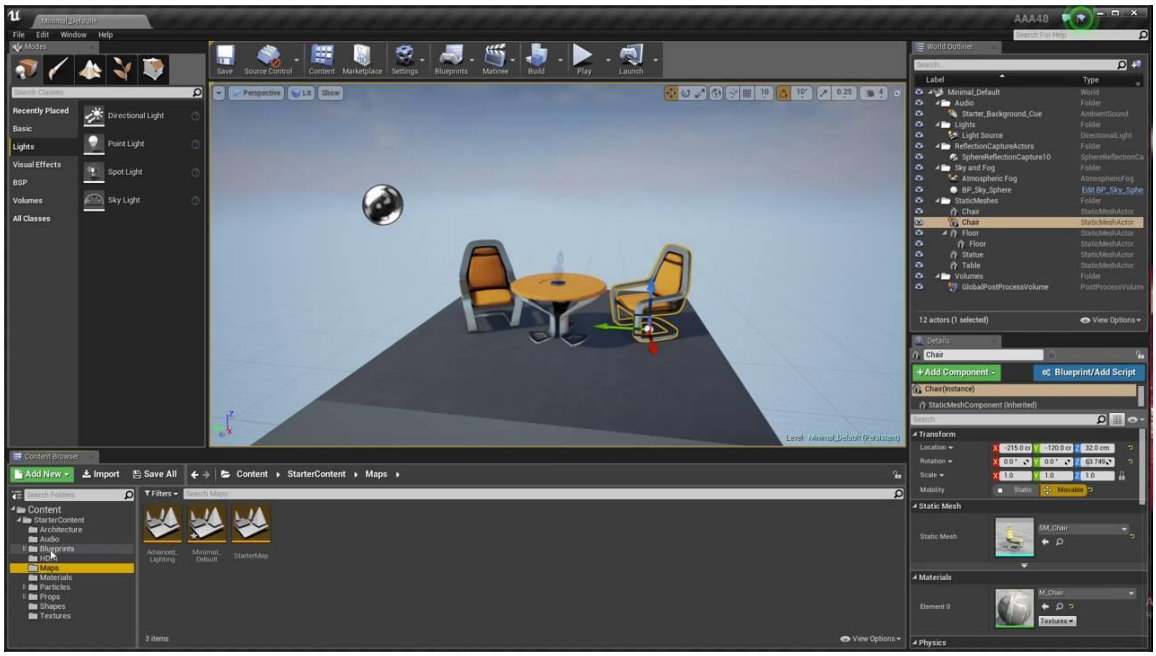


Рис. 50. Интерфейс Unreal Engine 4

Unreal Engine 4 – это набор инструментов для разработки игр. Он имеет широкие возможности: от создания двухмерных игр на мобильные телефоны до проектов для консолей. Разработка в Unreal Engine 4 очень проста для начинающих. С помощью системы визуального создания скриптов Blueprints Visual Scripting можно создавать готовые игры, не написав ни строчки кода. В сочетании с удобным интерфейсом это позволяет быстро изготавливать рабочие прототипы [45]. Для программирования используется язык C++ или встроенный редактор кода Blueprints Visual Scripting. Пример интерфейса программы показан на рис. 50.

1.5. Цифровой звук в системах виртуальной реальности

Первоначально запись звука для компьютерных систем осуществлялась с помощью кассет и грампластинок.

В 1970 г. появилась монофоническая цифровая запись. Цифровая звукозапись – технология преобразования аналогового звука в цифровой с целью сохранения его на физическом носителе для возможности последующего воспроизведения записанного сигнала.

Принцип цифрового представления колебаний звукозаписи достаточно прост: нужно преобразовать аналоговый сигнал в цифровой, это осуществляет устройство – аналого-цифровой преобразователь (АЦП); произвести сохранение полученных цифровых данных на носитель (магнитную ленту (DAT), жёсткий диск, оптический диск или флеш-память). Для того чтобы прослушать сделанную запись, необходимо воспроизведение сделанной записи с носителя и обратное преобразование из цифрового сигнала в аналоговый с помощью цифро-аналогового преобразователя (ЦАП).

Принцип действия АЦП заключается в том, что аналоговый сигнал, полученный от микрофонов и электромузыкальных инструментов, преобразовывается в цифровой. Это преобразование включает в себя следующие операции.

- Ограничение полосы частот производится при помощи фильтра нижних частот для подавления спектральных компонент, частота которых превышает половину частоты дискретизации.
- Осуществляется дискретизация во времени, то есть происходит замена непрерывного аналогового сигнала последовательностью его значений в дискретные моменты времени – отсчетов. Эта задача решается путём использования специальной схемы на входе АЦП, которая представляет собой устройство выборки-хранения.

- Квантование по уровню представляет собой замену величины отсчета сигнала ближайшим значением из набора фиксированных величин – уровней квантования.
- В результате кодирования или оцифровки, значение каждого квантованного отсчета представляется в виде числа, соответствующего порядковому номеру уровня квантования (рис. 51).

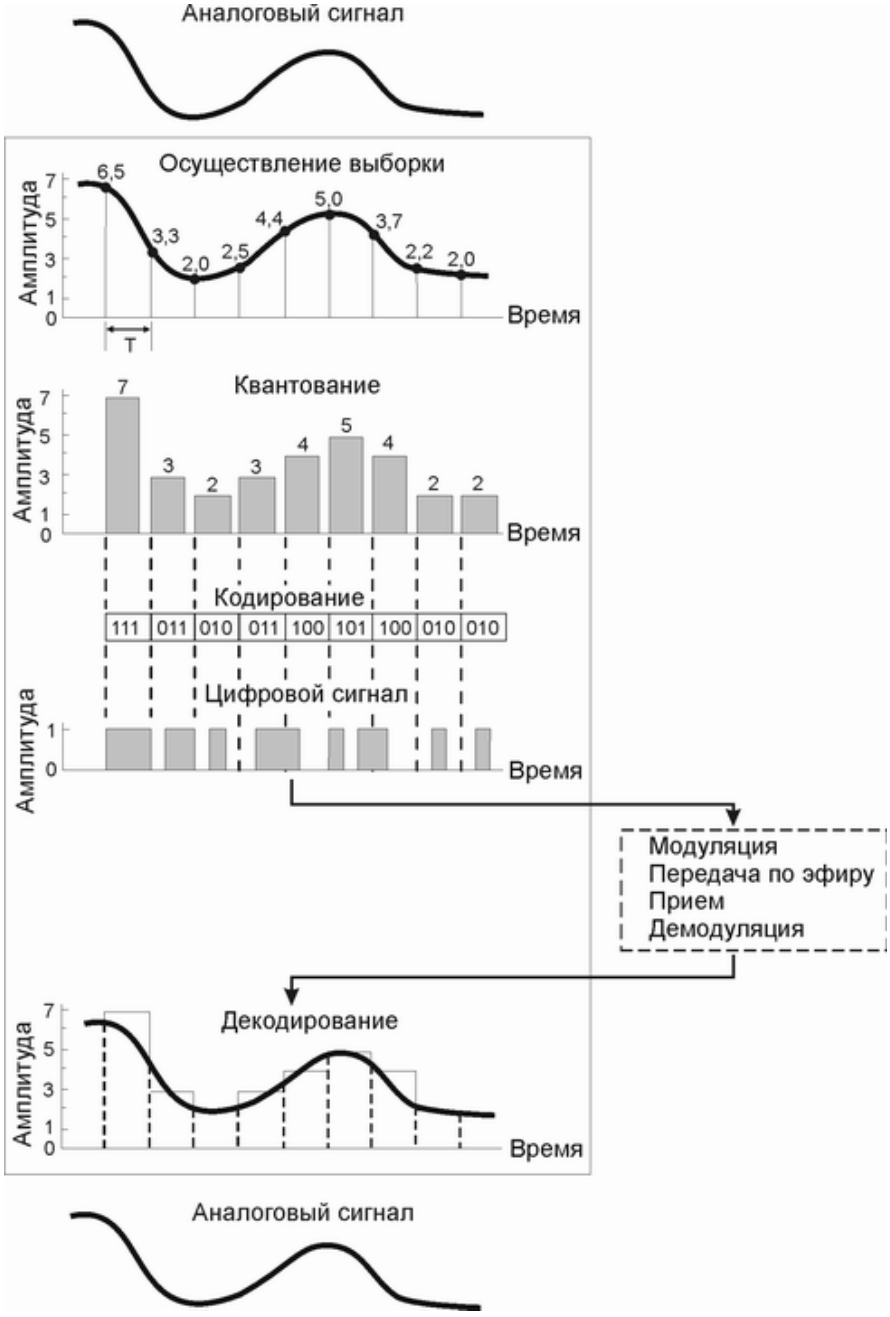


Рис. 51. Преобразование звука из аналоговой в цифровые формы

Спустя 10 лет существования монофонической цифровой записи японская компания Famicom Dendy, занимающаяся разработкой игровых приставок, стала использовать в своих устройствах 5 канальный цифровой звук (1980 г.).

В 1985 г. появились 8-битные цифро-аналоговые преобразователи, табличный синтез звука.

Запись цифрового звука в настоящее время осуществляется на студиях звукозаписи, под управлением персональных компьютеров и другой дорогостоящей и качественной аппаратуры. Также довольно широко распространено явление «домашней студии», в которой применяется профессиональное и полупрофессиональное звукозаписывающее оборудование, позволяющее создавать качественную запись в домашних условиях.

Применяются звуковые карты в составе компьютеров, которые производят обработку звука в своих АЦП и ЦАП. Чаще всего обработка осуществляется в 24 битах и 96 кГц, дальнейшее повышение битности и частоты дискретизации практически не улучшает качества записи.

Существует целый класс компьютерных программ – звуковых редакторов, которые позволяют работать со звуком следующим образом:

- записывать входящий звуковой поток;
- создавать (генерировать) звук;
- изменять существующую запись (добавлять семплы, изменять тембр, скорость звука, вырезать части и т.п.);
- перезаписывать из одного формата в другой;
- конвертировать разные аудиокодеки.

Некоторые простые программы, позволяют осуществлять только конвертацию форматов и кодеков.

На сегодняшний день в процессе звукозаписи широко применяются сэмплы. Это относительно небольшой оцифрованный звуковой фрагмент. Семплы широко используются при написании современной музыки. На сегодняшний день существует огромное количество всевозможных семплеров, ромплеров и подобных устройств, которые значительно оптимизируют работу с семплами.

Объемный звук или его еще называют 3D-звук (surround sound), появился в 1991 г. Его воспроизведение основано на принципе бинауральной записи. Бинауральная запись – метод звуковой записи, при котором используется специальное расположение микрофонов, предназначенное для последующего прослушивания через наушники. Обычно при этом методе записи используется специальный манекен, повторяющий анатомическое строение человеческой головы (иногда вплоть до ушей) (рис. 52). Строение внешнего уха у каждого из нас индивидуально, и человек с раннего детства привыкает слышать окружающий мир с таким строением уха. Если строение уха модели отличается от строения уха слушателя, то это может привести к искаженному восприятию записи [40, 41].

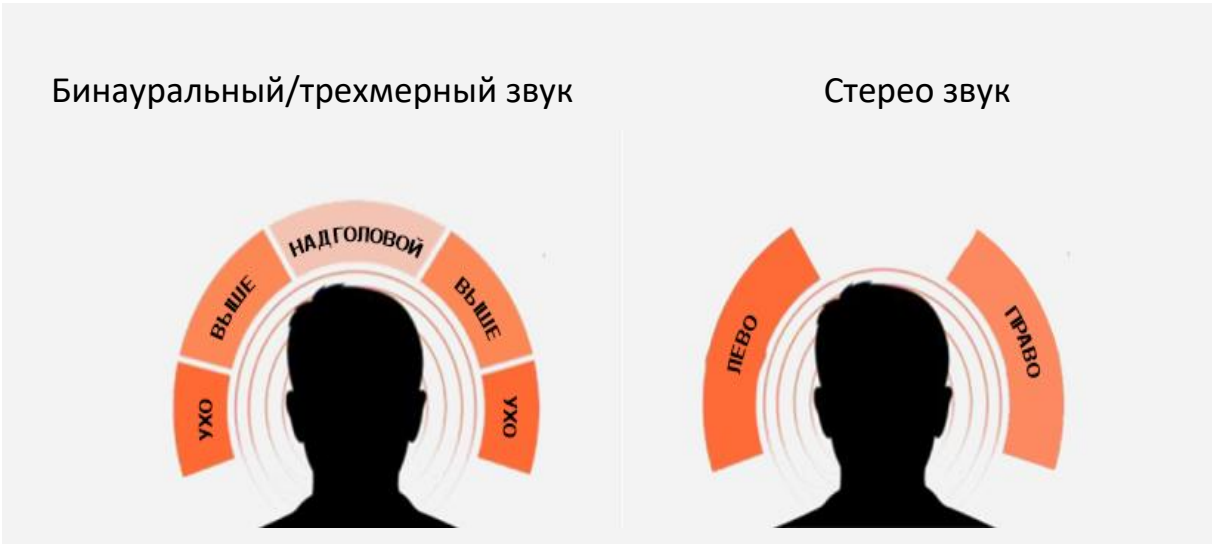


Рис. 52. Отличия 3D звука от стерео

1.6. Физика в виртуальной реальности

Материал параграфа подготовлен на основе книги А. Ламота «Программирование игр для Windows. Советы профессионала» [28].

Для создания физических воздействие в системах виртуальной реальности используется математическое моделирование. Большинство VR-систем используют модели на базе стандартной классической физики Ньютона, которая вполне справедлива для описания движения объектов в разумных пределах масс, размеров и скоростей (т.е. скорость должна быть гораздо меньше скорости света, объекты – гораздо больше отдельного атома, но гораздо меньше галактики). Но даже такое моделирование на основе физики Ньютона требует немалых затрат компьютерных сил. Простая имитация дождя или бильярда, выполненная полностью корректно, потребует высокой мощности, что требует высокопроизводительного «железа».

Тем не менее моделирование и дождя, и бильярда можно воспроизвести практически на любом современном компьютере. Так в чем же дело? Программисты этих игр отдавали себе отчет в том, что физика, которую они должны смоделировать, слишком сложна и следует ограничиться очень приближенными моделями, дающими результаты, близкие к тем, с которыми игрок сталкивается в реальной жизни. В таких программах множество различных уловок, оптимизаций и упрощений моделируемой системы. Например, гораздо проще вычислить результат столкновения двух сфер, чем результат столкновения астероидов неправильной формы, поэтому программист может аппроксимировать астероиды в игре простыми сферами.

Рассмотрим некоторые базовые концепции и свойства пространства, времени и материи.

В виртуальной реальности концепция **массы** используется в большинстве случаев абстрактно, как относительная величина. Например, можно указать, что масса корабля – 100 единиц, а масса астероида – 10 000. Можно использовать для описания массы килограммы, но пока не начинается реальное физическое моделирование, никакого значения это не имеет. Необходимо знать лишь то, что объект массой 100 единиц в два раза массивнее объекта с массой 50 единиц.

Время (упрощенно 1ед. = 1кадр). Время в реальной жизни измеряется в секундах, минутах, часах и т.д. Если нужны более точные измерения, можно вычислять время в миллисекундах (1 ms = 0,001 с), микросекундах (1 μs = 10^{-6} с) и прочих долях секунды. Однако в большинстве VR-систем никакой корреляции между реальным временем и временем в игре не наблюдается. Разрабатываемые алгоритмы в большей степени привязаны к частоте кадров, чем к реальному времени и секундам (за исключением систем, где моделируется реальное время). В большинстве VR-систем один кадр – это и есть одна виртуальная секунда, другими словами, наименьшее различимое количество времени.

Положение в пространстве (расчет центра масс). Каждый объект имеет определенное положение. x, y, z – в трехмерном пространстве; x, y – в двухмерном, x (иногда используется обозначение s) – в одномерном. Однако иногда непонятно, что считать положением объекта, даже когда известно, где именно он находится.

Например, выбирая точку, которая определяет положение сферы, вы наверняка выберете ее центр. Но что делать, если этот объект, например, молоток? Молоток представляет собой объект сложной формы, так что большинство физиков будут использовать его центр масс в качестве характеристики его положения в пространстве (рис. 53).

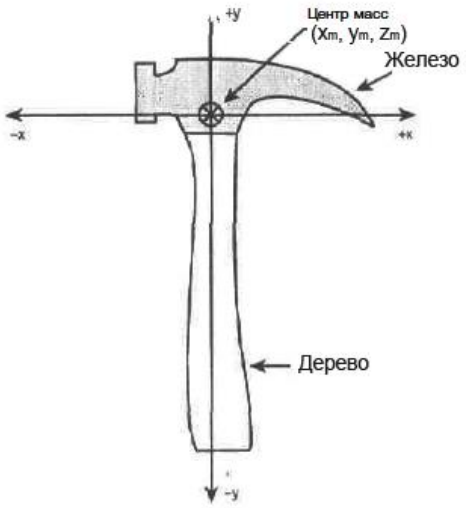


Рис. 53. Центр масс молотка

Однако в виртуальной реальности концепция **положения в пространстве** гораздо проще. Большинство программистов просто описывают вокруг объекта прямоугольник, окружность (параллелепипед, сферу, в зависимости от размерности игры), и используют центр описанного объекта в качестве центра рассматриваемого объекта. В большинстве случаев такое описание положения объекта вполне приемлемо, особенно когда основная масса объекта расположена в его центре. Вычислить центр масс достаточно просто, нужно разбить объект на элементарные малые по размеру ячейки и вычислить их вес (например, как количество точек объекта, попадающих в данную часть), а затем вычислить координаты центра масс по общей формуле:

$$\overrightarrow{r_c} = \frac{\sum_{i=1}^{n} \vec{r}_i m_i}{\sum_{i=1}^{n} m_i} \quad .$$

Скорость представляет собой мгновенную меру поступательного движения объекта и измеряется как отношение пройденного объектом расстояния ко времени, за которое оно было пройдено. Единица измерения скорости – m/s. Математически скорость представляет собой первую производную по времени радиус вектора движущейся точки:

$$\vec{v} = \frac{d\vec{r}}{dt} \quad .$$

Объект в виртуальной реальности перемещается со скоростью, измеряемой в пикселах в секунду.

Ускорение похоже на скорость, но определяет скорость изменения скорости. На рис. 54 проиллюстрированы объект, движущийся с постоянной скоростью, и объекты, движущиеся ускоренно. Графиком зависимости скорости от времени для объекта, движущегося с постоянной скоростью, будет горизонтальная прямая. Для объекта, движущегося с ускорением, зависимость скорости от времени представляет собой наклонную прямую (при постоянном ускорении) или кривую.

Математически ускорение определяется так же, как и скорость, с той разницей, что радиус-вектор объекта заменяется вектором его скорости:

$$\vec{a} = \frac{d\vec{v}}{dt} \quad .$$

Одной из важнейших концепций физики является **сила**. На рис. 55 показан один из вариантов иллюстрации силы. На нем изображен объект с массой m, находящийся на плоскости в поле тяжести Земли, ускорение свободного падения в котором обозначается как *g*.

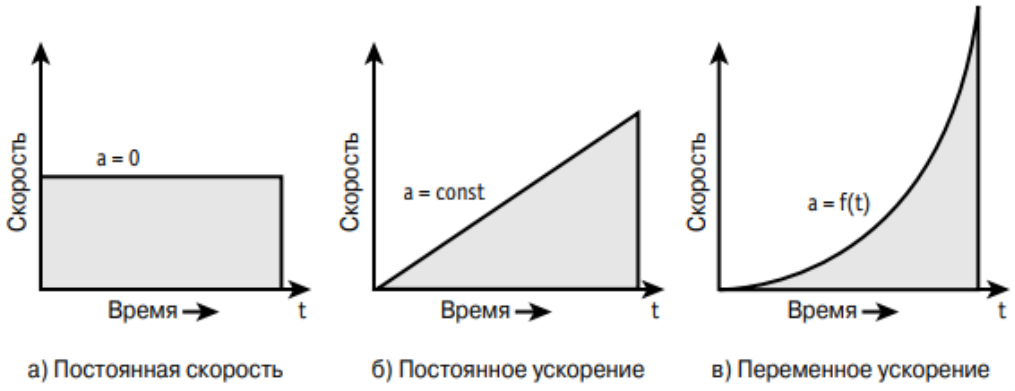


Рис. 54. Графики изменения скорости при различных ускорениях

Связь между силой, массой и ускорением описывается вторым законом Ньютона: $F=ma$. Другими словами, сила, прилагаемая к объекту для его движения с ускорением *a*, равна произведению массы на это ускорение.

Второй закон Ньютона может быть переписан в другом виде: $a = F/m$, что определяет ускорение движения тела под действием силы.

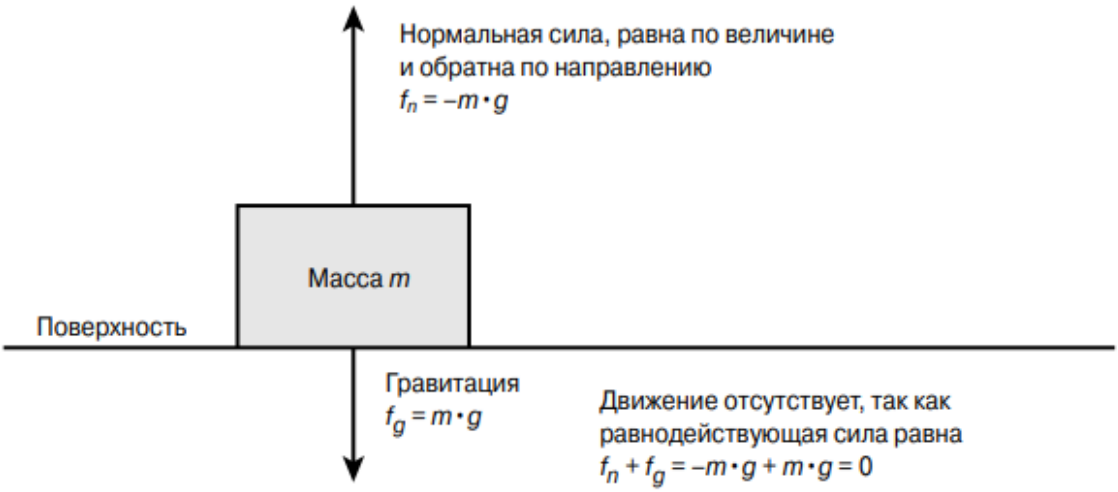


Рис. 55. Сила и вес

Концепция силы в VR-системах используется в следующих случаях:

- Нужно применить силу к некоторому объекту и вычислить получившееся ускорение.
- Два объекта сталкиваются, и нужно вычислить силы, действующие на каждый из объектов.
- Пушка в игре использует одинаковый пороховой заряд, но ядра разной массы, и требуется рассчитать ускорения ядер для определения дальности стрельбы.

Импульс – это свойство движущегося объекта, которое определяется как произведение массы объекта на скорость движения: $P=mv$. Очевидно, что единица измерения импульса – $kg \cdot m/s$. Импульс используется в законе сохранения энергии, с помощью которого можно вычислить скорости тел при упругом соударении. Например, при соударении тел массами 2 и 3 кг, движущимися со скоростями соответственно 4 и -2 м/с, их скорости после абсолютно упругого соударения станут равны соответственно -3,2 и 2,8 м/с.

Гравитация – это сила, которая притягивает любой объект во Вселенной к другим объектам. Эта сила невидима и в отличие от электромагнитного поля не может быть экранирована. В действительности гравитация является не силой, а следствием искривления пространства при наличии массы (рис. 56).

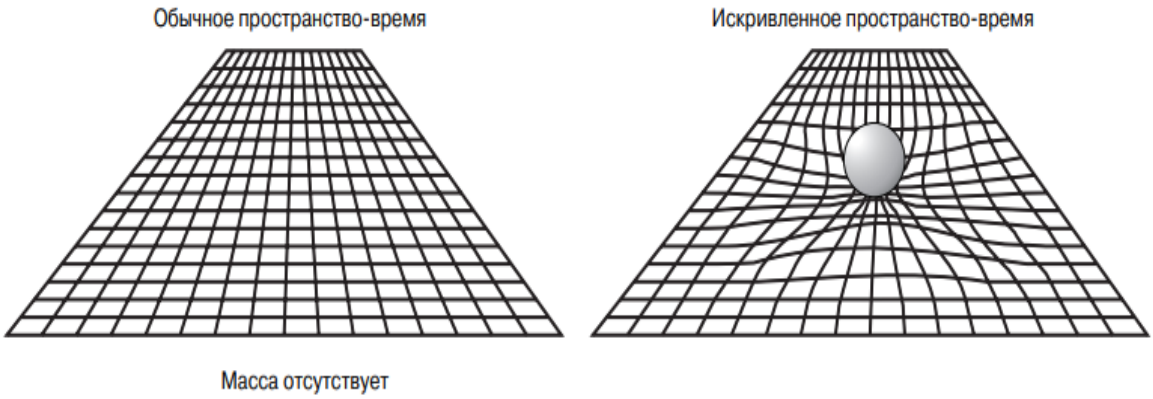


Рис. 56. Гравитация и пространство-время

Сила гравитационного притяжения между двумя телами с массами m_1 и m_2 равна:

$$F=\gamma\frac{m_1*m_2}{r^2} \quad .$$

Ускорение, вызванное гравитационным притяжением, постоянно для всех падающих тел независимо от их массы и равно примерно 9,8 м/с^2.

Падающие объекты вполне интересны, но в виртуальной реальности есть еще более интересные моменты, например, стрельба из пушки. На рис. 57 в графическом виде показана задача расчета баллистической траектории. Пусть имеется плоскость $y = 0$, танк с координатами (0,0) с пушкой, направленной под углом ϑ к оси X. Что произойдет, если выстрелить снарядом массы m со скоростью v_i? Как далеко улетит снаряд? Сопротивление воздуха отсутствует.

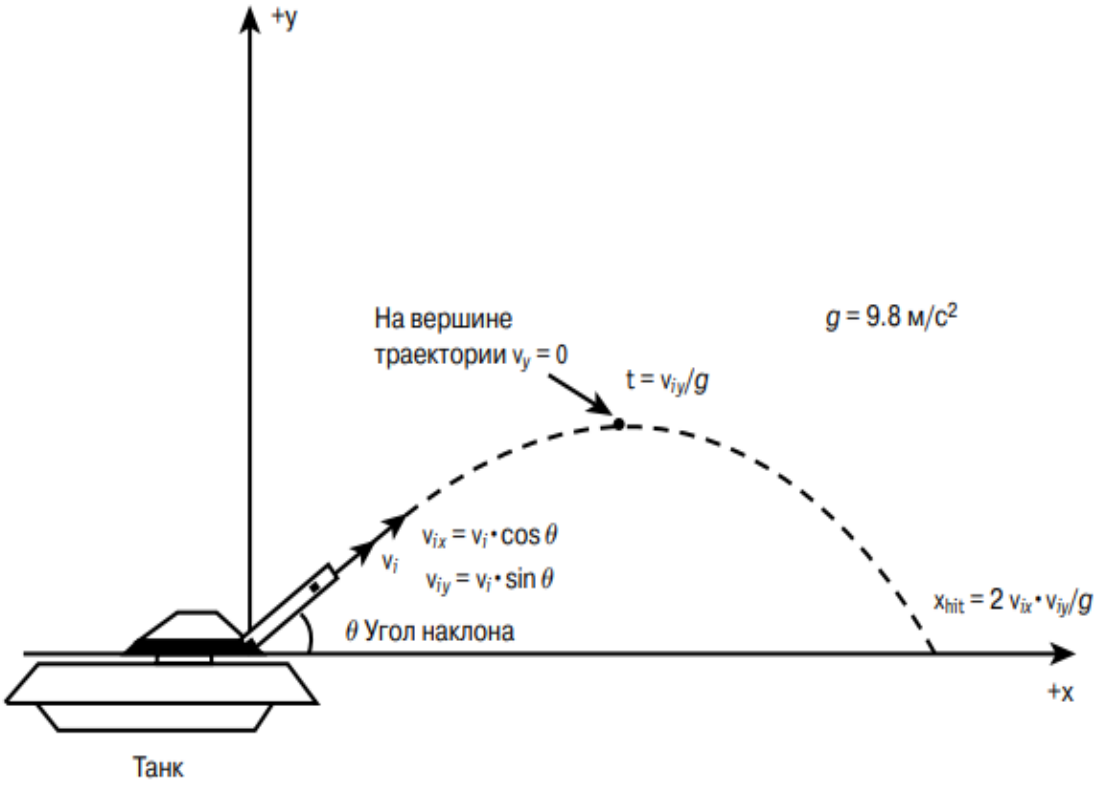


Рис. 57. Определение траектории движения снаряда

Следующая физическая величина, которую стоит рассмотреть – **трение**. Трением, по сути, является любая сила, которая замедляет движение или поглощает энергию из другой системы. Например, в автомобиле 30–40% вырабатываемой двигателем энергии уходит в тепло или затрачивается на механическое трение. С этой точки зрения самым эффективным средством передвижения, пожалуй, является велосипед, потери которого составляют не более 10–20%.

Трение представляет собой сопротивление в направлении, противоположном направлению движения, и, следовательно, может быть смоделировано при помощи использования силы, которую обычно называют силой трения. На рис. 58 изображена стандартная модель трения тела с массой *m* на плоскости.

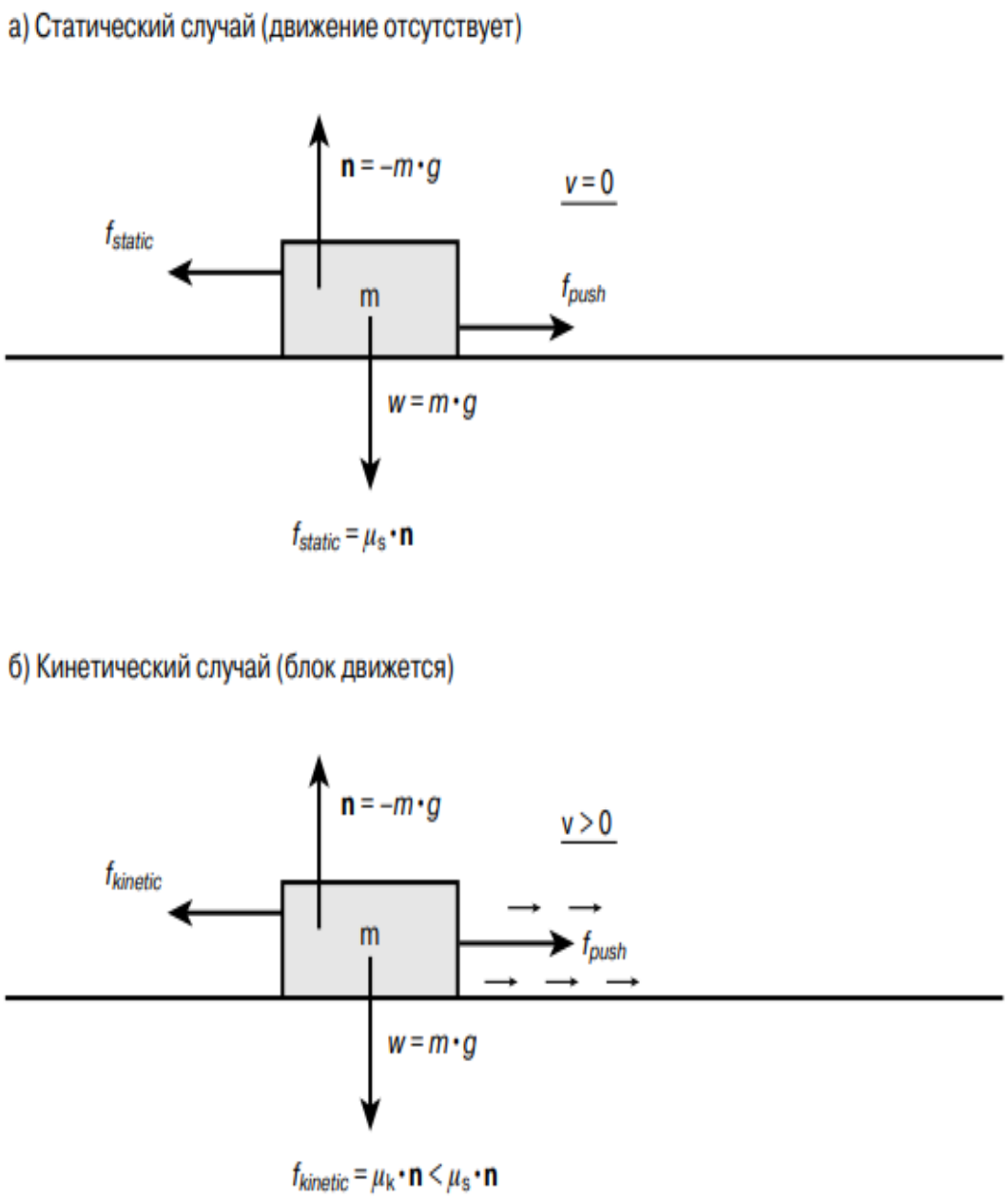


Рис. 58. Модель трения

Трение вычисляется по формуле

$$F_{mp}=mg\mu$$

Где m – масса, g – УСП, μ – статический коэффициент трения.

Определения **столкновения** или **коллизии** [19] – еще одна интересная задача при моделировании физики. Существует два типа столкнове-

ний: упругие и неупругие. При упругих столкновениях выполняются законы сохранения импульса и кинетической энергии; при неупругих – часть кинетической энергии переходит в тепловую. В большинстве систем ВР неупругие столкновения не рассматриваются в силу сложности их моделирования; более того, зачастую используемые для моделирования столкновений средства не имеют ничего общего с физикой соударений. Чтобы упростить сложные вычисления направлений векторов и скоростей, вместо объекта неправильной формы берут в качестве объекта коллизии сферу или параллелепипед (рис. 59). Такое действие называют упрощением.

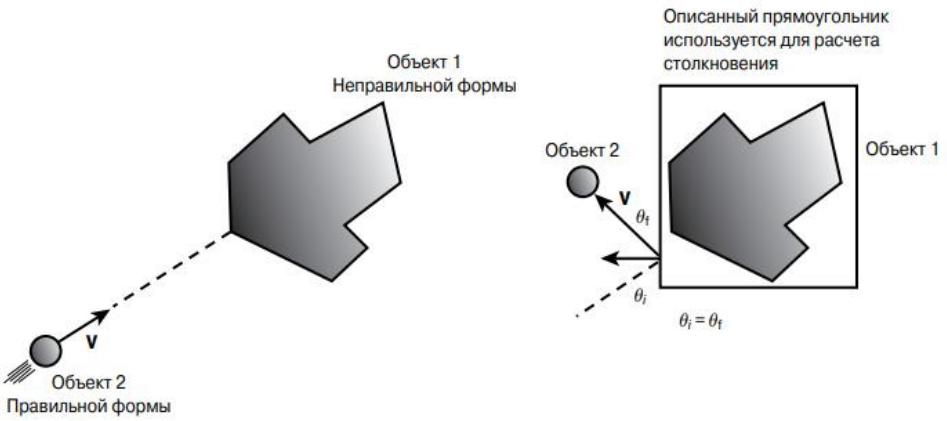


Рис. 59. Упрощение столкновений объектов

Тема физического моделирования рассматривается в компьютерном моделировании. Каждая модель, например: модель маятника, движения тела в среде с сопротивлением, упругое падение мяча, баллистическая траектория и др. – может быть смоделирована в системах моделирования без программирования. Например, модель движения тела в среде с сопротивлением строится в системе RMD за несколько минут, при этом программирование знать не обязательно [25]. Результат такого моделирования показан на рис. 60.

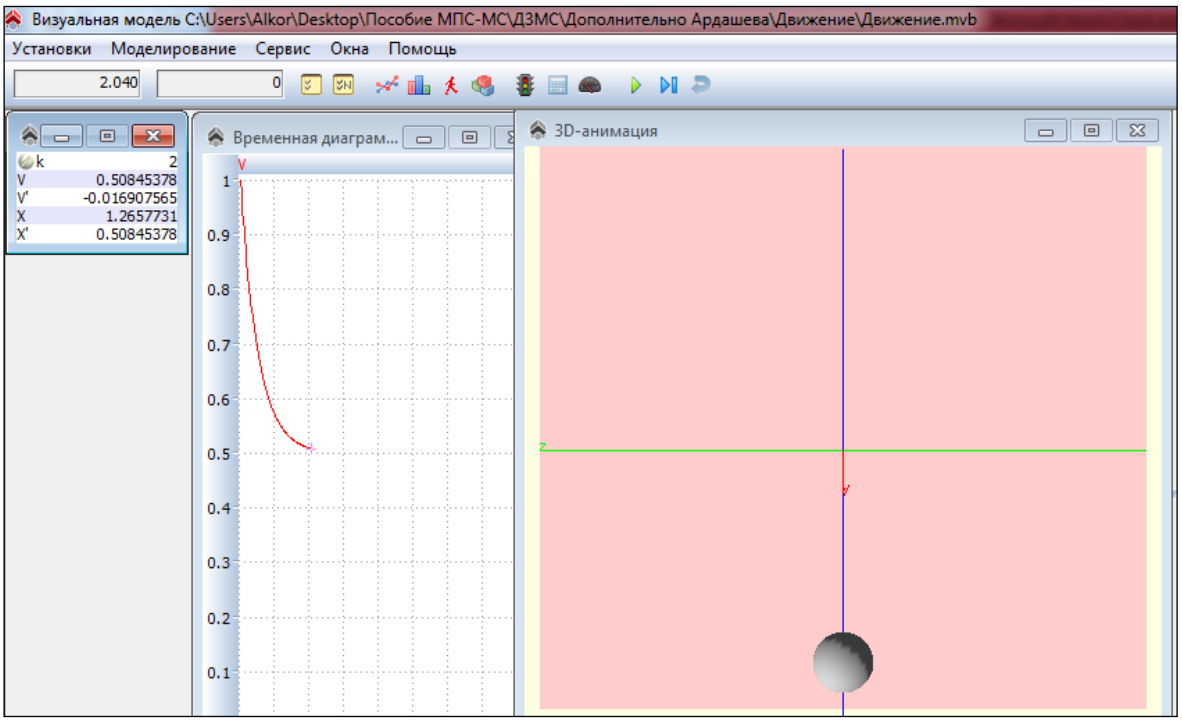


Рис. 60. Результаты моделирования движения тела (RMD)
в среде с сопротивлением

Тема физического моделирования требует от программиста навыков моделирования, математического анализа, понимания физики и др. компетенций [46]. Но в ускоряющемся мире информационных технологий на тщательную проработку этих компетенций нет времени. Именно поэтому появилась необходимость в создании специальных наборов функций, библиотек, с помощью которых создавать физику в ВР могли бы все желающие разработчики.

Рассмотрим некоторые из этих библиотек: PhysX, RenderWare Physics, Havok, ODE, Tokamak.

NVIDIA PhysX – самый популярный в мире движок физической симуляции. Подпрограммное обеспечение PhysX SDK позволяет разработчикам игр избегать написания собственного программного кода для обработки сложных физических взаимодействий в современных компьютерных играх. PhysX SDK может использоваться не только в среде Microsoft Windows, но и

в Linux, однако поддержка процессора PhysX пока работает только для Windows.

В отличие от большинства других физических движков, которые устанавливаются вместе с игрой, PhysX SDK необходимо установить отдельно. Он устанавливается как отдельный драйвер. Если на компьютере есть плата PhysX, то драйвер PhysX SDK при работе будет использовать её ресурсы. Если же PhysX отсутствует, то вычислительные задачи будут переноситься на центральный процессор [5].

Физический движок PhysX SDK состоит из трёх главных компонентов по обработке физики:

- обработка твёрдых тел (англ. rigid body);
- обработка тканей (англ. cloth);
- обработка жидкостей (англ. fluid).

На рис. 61 показан пример моделирования жидкости с использованием PhysX.

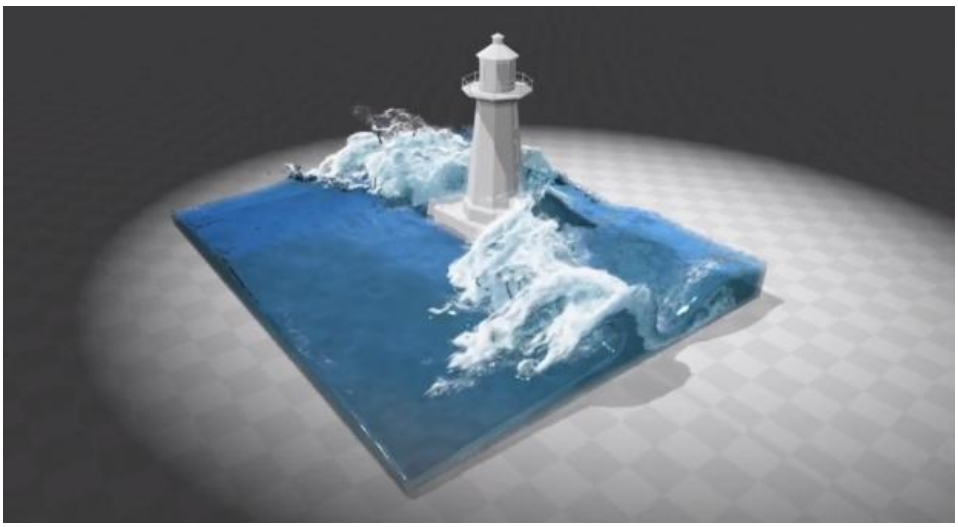


Рис. 61. Моделирование жидкости в PhysX

Havok Physics (более известный как просто Havok) – физический движок, разработанный ирландской компанией Havok. Движок создаёт симуляцию физического взаимодействия в реальном времени, что делает мир

игры более живым и реалистичным, подобно физике Ragdoll. Havok Physics является кросс-платформенным движком [5, 19].

Havok используется не только в играх, но и обучающих симуляторах, например в тренажере вождения 3D-инструктор, который предназначен для получения навыков вождения автомобиля и соблюдения правил дорожного движения. Havok также используется в продуктах компании Autodesk Media & Entertainment, программных пакетах 3DS Max и Maya (рис. 62).

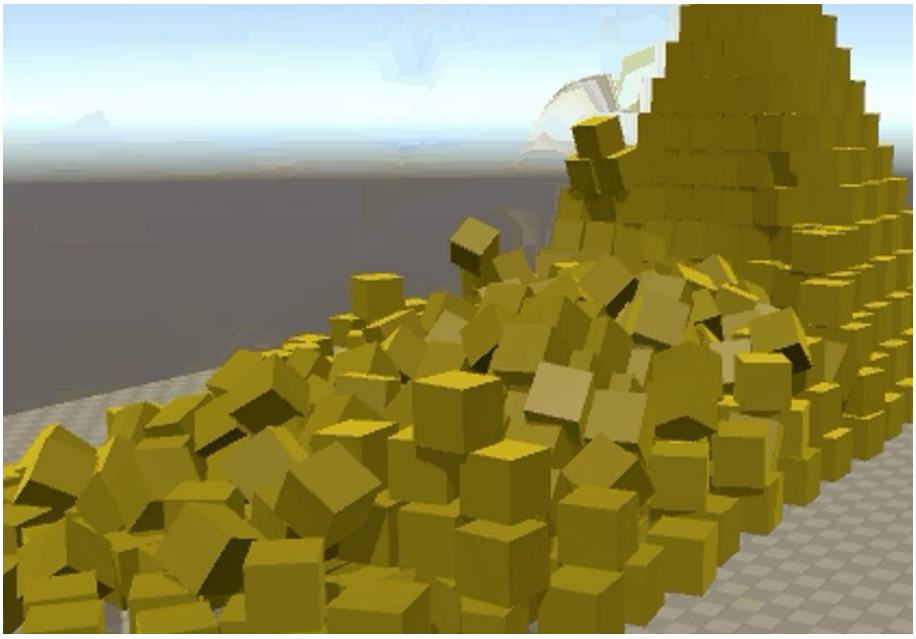


Рис. 62. Моделирование физики твердых тел в Unity 3D с использованием Havok

Tokamak Game Physics SDK (далее – Tokamak) – физический движок, работающий в режиме реального времени, который распространяется на основе свободной лицензии BSD с открытыми исходными кодами.

В начале Tokamak был свободным только для некоммерческого использования, однако с мая 2007 г. движок перешёл на открытые исходные коды под лицензией BSD.

Физический движок Tokamak обладает уникальным итерационным методом для обработки ограничений (англ. solving constraints). Этот метод

необходим для того, чтобы позволять разработчикам делать выбор между точностью и скоростью и обеспечивать более предсказуемое потребление ресурсов процессора и системной памяти физическим движком. Обработчик ограничений Tokamak не вовлекает в обработку большие матрицы, избегая таким образом ограничения на полосу пропускания памяти на некоторых игровых консолях.

Tokamak поддерживает множество типов связей (англ. joint), ограничений (англ. joint limits) и реалистическую модель трения. Tokamak оптимизирован для наложения (англ. stacking) большого количества объектов, что является часто требуемым многими разработчиками игр. Tokamak обеспечивает обнаружение столкновений (англ. collision detection) для примитивов (параллелепипед, сфера, капсула), комбинаций примитивов и произвольных статических полигональных сеток (англ. arbitrary static triangle meshes). Лёгкие «твёрдые частицы» обеспечивают эффекты частиц (англ. particle effects) в играх с минимальными затратами (рис. 63).

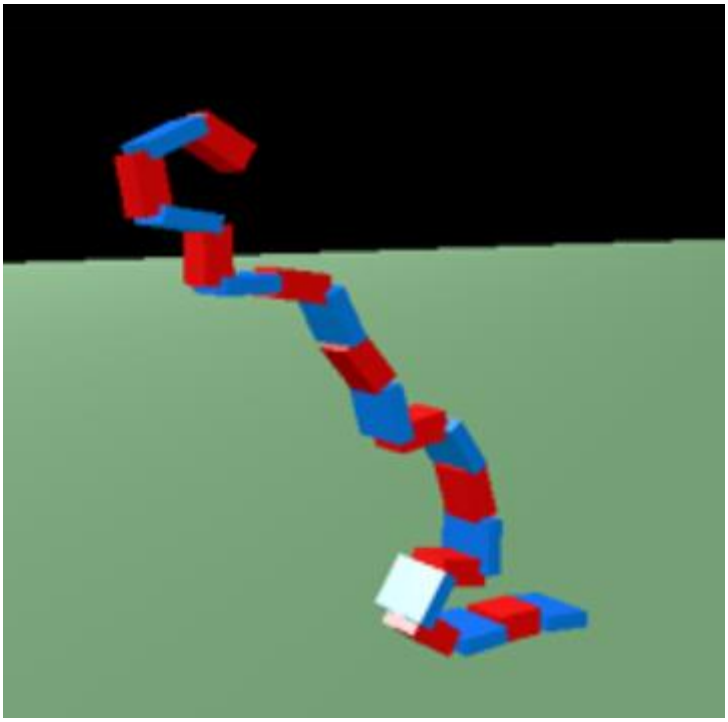


Рис. 63. Демонстрация физики «веревки», в которой используется шарнирное соединение Tokamak

Tokamak также поддерживает «разламываемую конструкторскую модель» (англ. Breakage Constructing models), которая разламывается в результате столкновений. Фрагменты (обломки, осколки) первоначальной модели будут созданы автоматически встроенными функциональными возможностями движка [7].

1.7. Искусственный интеллект в системах виртуальной реальности

В академическом смысле термин «искусственный интеллект» (ИИ) означает аппаратное или программное обеспечение, которое позволяет компьютеру «думать», т.е. обрабатывать информацию подобно человеку.

Системы виртуальной реальности просто немыслимы на сегодняшний день без искусственного интеллекта, ведь интерактивность нужно обрабатывать, следовательно, программировать реакцию на воздействие пользователей.

Каковы особенности естественного интеллекта?

- Способность решать не формализуемые или плохо формализуемые задачи (например, в искусстве или в быту).
- Способность обучаться эффективному решению новых задач (например, благодаря любознательности).
- Способность генерировать информацию (не передавать чужую информацию, а творить, создавать новую).
- Способность психологически адаптироваться (приспосабливаться) к среде в широком диапазоне условий.
- Способность классифицировать явления, события, ситуации, объекты.
- Способность к анализу (дедукции) и синтезу (индукции) как методам познания.

- Чувство юмора как способность находить противоречие в единстве, единство в противоречии и несхожести, как способность генерировать информацию из разнородных элементов.

Английский математик Алан Тьюринг придумал тест на интеллект «Computing machinary and intelligence» и опубликовал его в октябрьском номере журнала «Mind» в 1950. Испытуемый взаимодействует с компьютером и человеком. На основании ответов на вопросы испытуемый должен определить, с кем он разговаривает: с человеком или компьютерной программой. Задача компьютерной программы – ввести человека в заблуждение, заставив сделать неверный выбор.

Искусственный интеллект (англ. Artificial intelligence, AI) – наука и технология создания интеллектуальных машин, особенно интеллектуальных компьютерных программ. ИИ связан с задачей использования компьютеров для понимания человеческого интеллекта, но необязательно ограничивается биологически правдоподобными методами.

Творцами искусственного интеллекта в свое время являлись ученые из разных областей науки: математики, биологи, физиологи, психологи, лингвисты и др. Технологии искусственного интеллекта лежат на стыке наук.

В развитие искусственного интеллекта внесли свой вклад следующие ученые:

- Н. Винер (математик), У.Р. Эшби (биолог) – основоположники кибернетики, впервые заявили, что машины могут быть умнее людей, дали первоначальный толчок развитию теории искусственного интеллекта.

- У. Маккаллок, У. Питс (физиологи) в 1943 г. предложили формальную модель нейрона; основоположники нейрокибернетики.

- А. Тьюринг (математик) – в 1937 г. изобрел универсальную алгоритмическую «машину Тьюринга»; предложил интеллектуальный «тест Тьюринга», позволяющий определить, «разумна» ли машина в сравнительном диалоге с ней и разумным человеком.

• Дж. Фон Нейман (математик) – один из основоположников теории игр и теории самовоспроизводящихся автоматов, архитектуры первых поколений компьютеров.

• М. Мински (математик) – автор понятия «фрейма» основополагающего в машинном представлении знаний; один из авторов теории персептрона – устройства для распознавания образов.

• М. Сомальвико (кибернетик), А. Азимов (биохимик, писатель) – основоположники интеллектуальной робототехники.

• Г. Саймон, У. Рейтман (психологи) – авторы и разработчики первых лабиринтных интеллектуальных моделей, построенных на принципах эвристического программирования.

• Р. Беллман (математик), С.Ю. Маслов (логик) – авторы динамического подхода к лабиринтным интеллектуальным моделям (динамического программирования, обратного метода доказательств).

• Ф. Розенблатт (физиолог), М.М. Бонгард (физик) – первооткрыватели проблемы распознавания образов; разработчики устройств и моделей распознавания и классификации.

• Л. Заде, А.Н. Колмогоров, А.Н. Тихонов, М.А. Гиршик (математики)– авторы математических методов решения плохо формализованных задач и принятия решений в условиях неопределенности.

• Н. Хомски (математик, филолог) – основоположник математической лингвистики.

• Л.Р. Лурия (психолог) – основоположник нейропсихологии, изучающей глубинные механизмы познавательной деятельности мозга и других интеллектуальных функций мозга.

• К.Э. Шеннон (инженер-связист), Р.Х. Зарипов (математик) – авторы теории и моделей машинного синтеза музыки.

Основные проблемы искусственного интеллекта являются:

• представление знаний;

- моделирование рассуждений;
- интеллектуальный интерфейс «человек – машина», «машина – машина»;
- планирование целесообразной деятельности;
- обучение и самообучение интеллектуальных систем;
- машинное творчество;
- интеллектуальные роботы.

Интеллектуальный интерфейс представляет собой взаимодействие компонент, представленных на рис. 64.

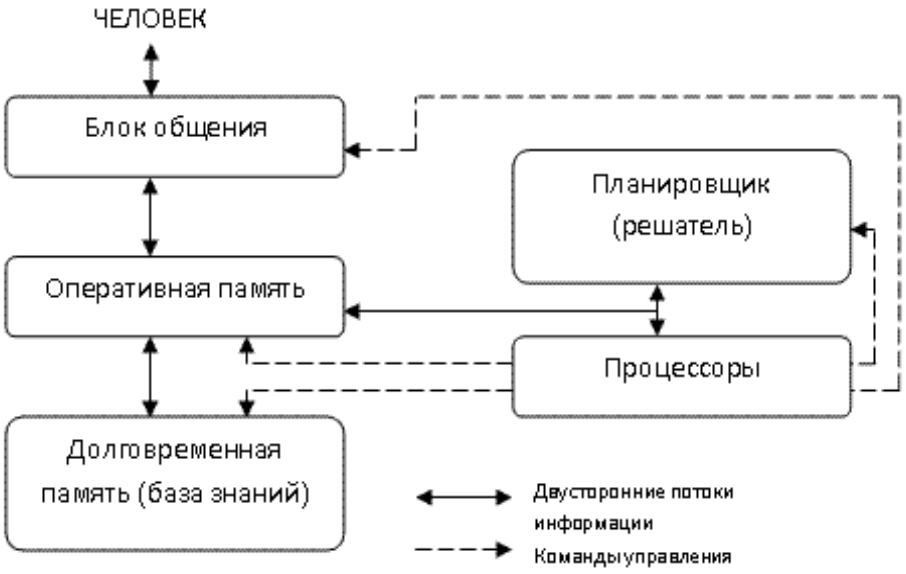


Рис. 64. Схема интеллектуального интерфейса

Существует понятие игрового искусственного интеллекта. Оно немного отличается от понятия искусственного интеллекта. Игровой искусственный интеллект (англ. Game artificial intelligence) – набор программных методик, которые используются в компьютерных играх для создания иллюзии интеллекта в поведении персонажей, управляемых компьютером. Игровой ИИ, помимо методов традиционного искусственного интеллекта, включает также алгоритмы теории управления, робототехники, компьютерной графики и информатики в целом.

Главная задача игрового интеллекта – не выиграть у игрока, а красиво ему проиграть. В играх, в которых важен творческий потенциал игрока, ИИ не может сражаться с человеком на равных. Чтобы уравнять шансы, применяют читерский, или обманный, искусственный интеллект.

Обманный искусственный интеллект компенсирует отсутствие стратегического мышления какими-либо другими преимуществами над игроком. Например, большее количество жизней, более быстрое передвижение или игнорирование тумана войны.

Пример игрового искусственного интеллекта. Если бот достаточно сильно отстаёт от основной массы гонщиков, он внезапно получает огромное увеличение скорости или другие параметры, позволяющие ему нагнать других гонщиков и снова стать конкурентоспособным соперником.

Еще в 1980-е гг. в процессе работы в сфере искусственного интеллекта стало понятно, что, скорее всего, искусственный разум подобный человеческому в обозримом будущем не создать. Слишком сложна задача, и вычислительные мощности малы. Вычислительные мощности сегодняшних суперкомпьютеров не справятся и с частью того количества сигналов, которые мгновенно передаются по нейронам в нашем мозге. Но не стоит опускать руки. Многочисленные «профиты» теории искусственного интеллекта уже сейчас широко применяются. Это анализ текстов в автоматических переводах; теория обучения нейронных сетей – в процессе анализа изображений и многое другое.

В последние годы, после временного затишья в 80-е гг. XX в., все только и говорят об искусственном интеллекте. Мы сталкиваемся с ним в поисковых системах, в банковском обслуживании, даже наложение фильтров на ваше фото может управляться искусственным интеллектом.

Рассмотрим наиболее типичные в системах виртуальной реальности задачи искусственного интеллекта, которые в настоящее время активно применяются:

- создание простых детерминированных алгоритмов;
- создание шаблонов и сценариев;
- управление поведением и состоянием;
- запоминание и обучение;
- создание деревьев планирования и принятия решений;
- поиск путей;
- создание языков сценариев;
- использование основ нейронных сетей;
- создание генетических алгоритмов;
- использование нечеткой логики.

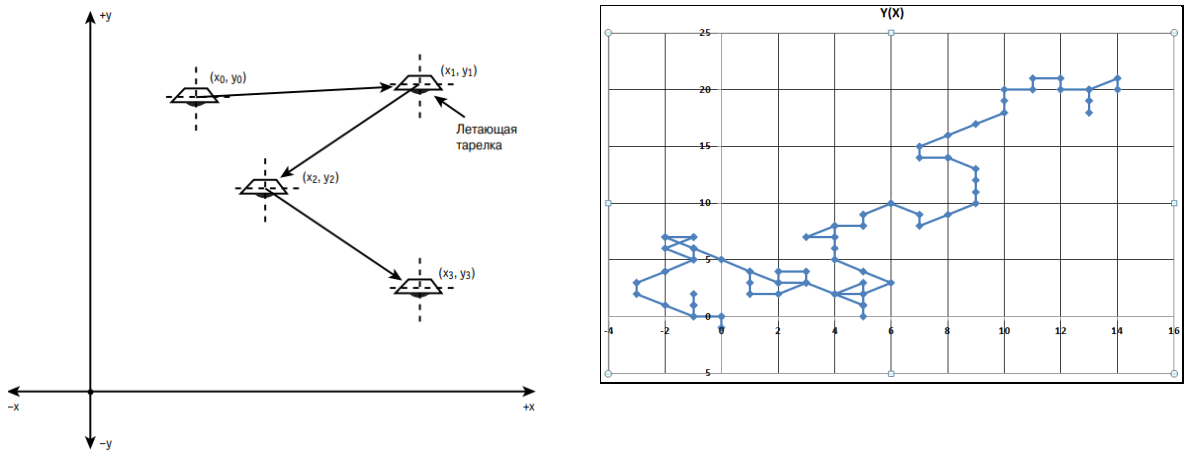
Детерминированные алгоритмы – это простые предсказуемые действия персонажей VR-систем, например: траектория движения птицы, маршрут неуправляемого или управляемого искусственным интеллектом транспортного средства и др. К детерминированным алгоритмам относят случайное блуждание, преследование и уклонение персонажа.

Алгоритм случайного движения, построенный на генерации случайных чисел (рис. 65):

```
fly_x0 := -8+random(16);
fly_y0 := -8+random(16);
```

Затем объект некоторое время движется с вычисленной скоростью:

```
// Движение с заданной скоростью в течение 10 циклов
For i := 0 to 9 do
begin
fly_x := fly_x + fly _x0;
fly_y := fly_y + fly_y0;
// ... Вычисление новой скорости объекта ...
End;
```



а) б)

Рис. 65. Случайное блуждание: для игрового объекта (а); модель, построенная в MS Excel (б) [25]

Случайным образом задается значение и вектор скорости, затем циклически (цикл на некоторое время) объект будет двигаться с заданной скоростью.

Хотя случайное перемещение и является совершенно непредсказуемым, применяется оно не так часто. Следующим шагом в развитии ИИ являются алгоритмы, которые учитывают состояние внешней среды и реагируют на него. В качестве примера рассмотрим алгоритм следования за объектом (tracking algorithm). ИИ на основе информации о положении некоторого (отслеживаемого) объекта изменяет траекторию нашего объекта таким образом, чтобы он двигался по направлению к отслеживаемому. Движение может быть направлено как непосредственно на отслеживаемый объект, так и (в более реалистичной модели) по некоторой кривой, направленной к объекту (наподобие траектории самонаводящейся ракеты), что показано на рис. 66.

Объект-преследователь сравнивает свои координаты в виртуальном мире с координатами объекта-жертвы и корректирует свои координаты таким образом, чтобы приблизиться к жертве. В простейшем случае преследование осуществляется на открытом пространстве.

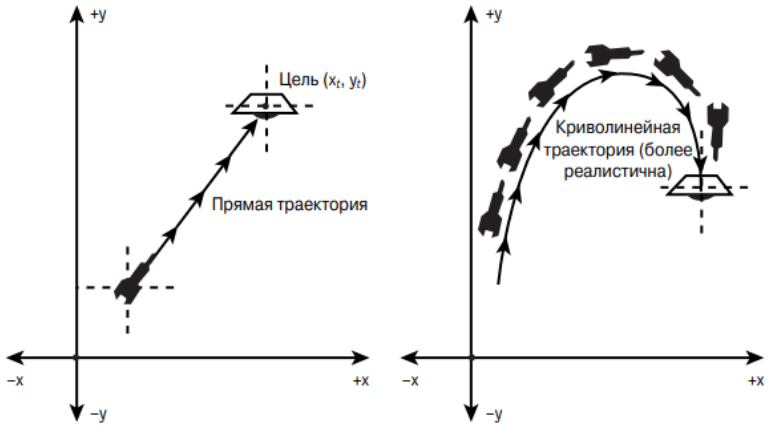


Рис. 66. Методы отслеживания перемещений

Преследование

```
// Предположим, что px,py - координаты игрока,
// ex,ey - координаты противника-преследователя
while ( игра )
begin
программный код
......
// Вначале рассматриваем перемещение по горизонтали
(ось X)
if ex > px then ex=ex-1
if ex < px then ex=ex+1
// Теперь рассматриваем вертикальный (Y) компонент
if ey > py then ey=ey-1
if ey < py then ey=ey+1
.... программный код
End;
```

Уклонение

```
//Предположим, что px,py – координаты игрока, ко-
торый убегает.
// ex,ey – координаты противника.
while (игра)
 {
игровой код
……
```

```
// перемещение по горизонтали
if (ex>px) then px := px - 1;
if (ex<px) then px := px + 1;
// перемещение по вертикали
if (ey>py) then py := py - 1;
if (ey<py) then py := py + 1;
……
игровой код
}
```

От простейших детерминированных алгоритмов перейдем к работе с шаблонами и сценариями.

Детерминированные алгоритмы вполне достаточны для решения множества задач, тем не менее довольно часто требуется создать объект, который действует по некоторому сценарию.

Приблизительный сценарий действий человека при поездке в автомобиле:

1. Вынуть ключи из кармана.
2. Открыть дверцу машины с помощью ключа.
3. Сесть в автомобиль.
4. Закрыть дверцу.
5. Вставить ключ в замок зажигания.
6. Повернуть ключ.
7. Завести двигатель.

Словом, следует выполнить целый ряд последовательных действий, повторяющихся каждый раз при необходимости куда-то ехать. Конечно, если что-то пойдет не так, последовательность может быть изменена. Например, если водитель оставил автомобиль в гараже не запертым или если у автомобиля сел аккумулятор. Шаблоны представляют собой важную часть интеллектуального поведения, включая поведение, присущее человеку.

```
// Предположим, что pattern - это массив, содержащий набор команд
// для реализации десяти различных шаблонов поведения
while (идет игра)
{
 ....
 код программы
 // Проверяем, закончена ли обработка текущего шаблона
 if (если обработка команд текущего шаблона закончена)
 {
 // Выбираем новый шаблон
 current_pattern = pattern[rand()%10];
 позиция противника = старая позиция + следующий элемент текущего шаблона
 Увеличим на единицу значение индекса элементов шаблона
 ....
 код программы
}
```

Тривиальный шаблон, который соответствует движению по квадрату (пример на С)

```
// Указывает на первую инструкцию.
// Каждая инструкция состоит из двух целых чисел,
int inrstruction_ptr = 0;
// Получаем число циклов
int cycles = square_stop_spin [instruction_ptr + 1];
// Обрабатываем инструкцию
 switch (square_stop_spin[instruction_ptr])
 {
```

```
case GO_FORWARD: // Перемещаем объект вперед
break;
case GO_BACKWARD: // Перемещаем объект назад
break;
case TURN_RIGHT_90: // Поворот объекта вправо
break;
case TURN_LEFT_90: // Поворот объекта влево
break;
case SELECT_RANDOM_DIRECTION:
// Выбор случайного направления
break;
case STOP: // Остановить объект
break;
} // switch
// Перемещаем указатель инструкций (на 2 числа)
instnjction__ptr+= 2;
// Проверяем, не добрались ли до конца программы
if (instruction_ptr > num_instructions * 2}
{
// Программа завершена
}
```

С помощью такой простой детерминированной технологии как шаблоны можно реализовать сложные модели поведения в VR-системах:

- шаблоны движения;
- шаблоны управления анимацией;
- выборы инструментария для работы в среде и пр.

Шаблон можно менять в зависимости от условий.

Например, если расстояние до противника слишком велико, то можно применить другой шаблон (рис. 67).

```
if (instruction[pattern++] == TEST_DISTANCE)
{
```

```
// Получаем значение операнда инструкции TEST_DIS-
TANCE
// и расстояние от игрока до объекта и сравниваем
их
int min_distance = instruction[pattern++];
if (Distance(player,object) > min_distance)
{
// Изменяем внутреннее состояние системы
// искусственного интеллекта...
ai_State = TRACK_PLAYER;
// ...или переключаемся на выполнение другого шаб-
лона
}
}
```

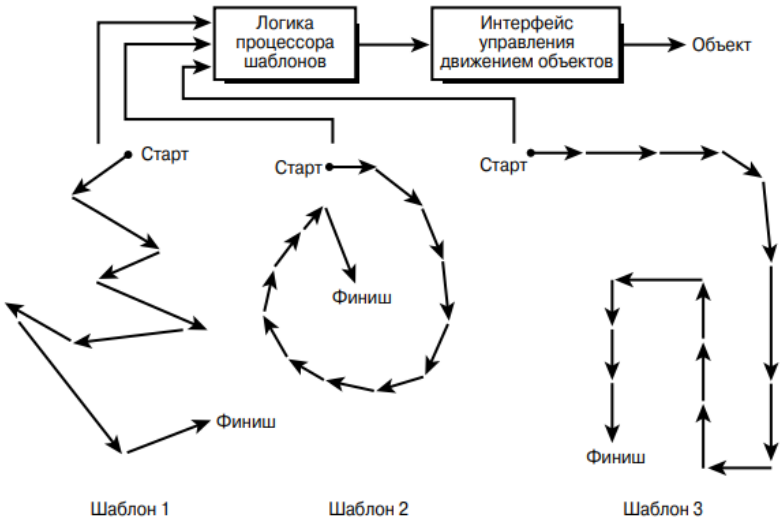


Рис. 67. Работа с шаблонами

Для создания устойчивого конечного автомата требуется разумное количество состояний, каждое из которых представляет различные цели или мотивы; входная информация для конечного автомата, такая как состояние среды и состояние других объектов в ней.

Например, простой персонаж игры может иметь несколько состояний.

- Состояние 1: Двигаться вперед.
- Состояние 2: Двигаться назад.
- Состояние 3: Поворот.
- Состояние 4: Останов.
- Состояние 5: Атака.
- Состояние 6: Погоня за игроком.
- Состояние 7: Бегство от игрока.

Для конечных автоматов характерна графовая модель визуализации (рис. 68), в которой можно отобразить и подсостояния поведения объекта.

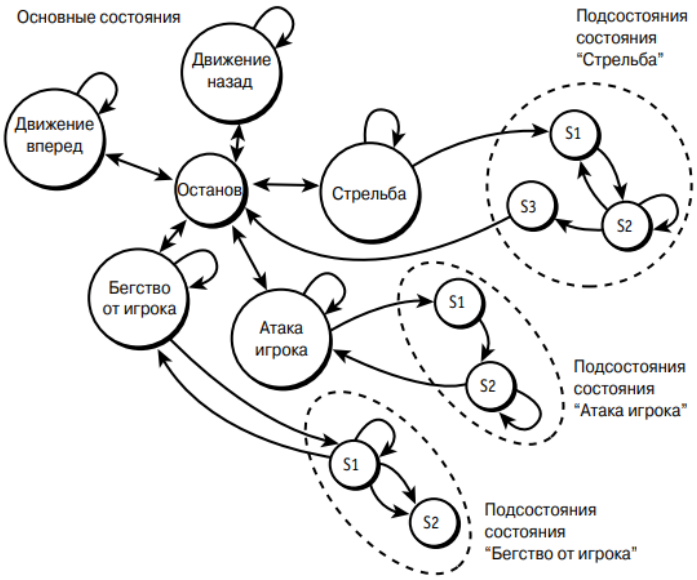


Рис. 68. Конечный автомат с подсостояниями

Поведение персонажей тесно связано с их индивидуальностью. Точнее, характер персонажа обусловлен его поведением, реакцией на то или иное состояние или событие. Индивидуальность, по сути, не что иное, как предсказуемость поведения того или иного персонажа. Наверняка, все помнят знаменитый психологический тест со шляпой, в котором по поведению человека можно определить его темперамент: сангвиник, флегматик, меланхолик или холерик. Получается, чтобы в системе виртуальной реально-

сти определенному персонажу придать индивидуальность, нужно определить, какое действие он будет совершать в зависимости от ситуации. Но если мы начнем программировать все известные состояния, то персонаж получится уж очень предсказуемым и неинтересным.

Приведем пример вероятностных характеристик двух персонажей в зависимости от тех или иных действий (рис. 69).

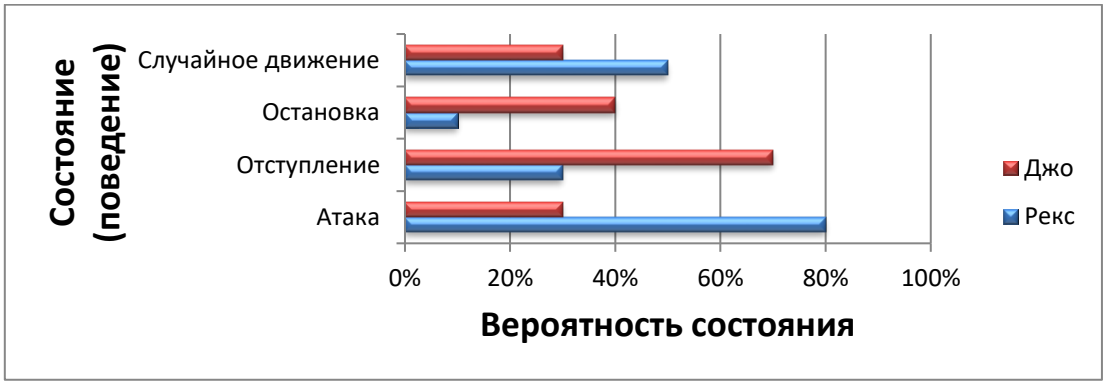


Рис. 69. Распределение вероятностей основных состояний разных персонажей игры

Вместо равновероятного выбора нового состояния, как было ранее, мы создаем и используем для каждого персонажа распределение вероятностей, определяющее его индивидуальность. Таким образом, можно сделать вывод, что персонаж Рекс более агрессивный и менее склонен отступать по сравнению с персонажем Джо. Если применить приведенное распределение вероятностей, Рекс будет бездумно бросаться в атаку, в то время как Джо предпочтет уклоняться от встречи с противником; кроме того, случайного в его поведении будет меньше, чем у Рекса, от которого можно ожидать чего угодно.

До этого момента рассматривался низкоуровневый ИИ, описанные выше технологии не являются реакционными и непосредственными. Высокоуровневая логика в них отсутствует. Высокоуровневый ИИ отличается планированием.

По сути, план представляет собой просто высокоуровневое множество действий, выполняемых в определенном порядке для достижения поставленной цели. Кроме действий, в план входит ряд условий, которые должны удовлетворяться перед тем, как будет выполнено то или иное конкретное действие.

Например, вот как может выглядеть план похода в театр:

1. Выбрать пьесу, которую бы вам хотелось посмотреть.
2. Отправиться в театр как минимум за час до начала спектакля.
3. Оказавшись в театре, приобрести билет.
4. Посмотреть пьесу и отправиться домой.

Выглядит более-менее разумно. Однако здесь опущено множество деталей. Например, в какой именно театр нужно ехать? Хватит ли одного часа, чтобы добраться до театра? Что делать, если не хватает денег на билет? Ну и так далее...

Эти детали могут быть существенны (или не существенны), в зависимости от того, насколько сложный план строится.

Однако должно быть столько условий и подпланов, чтобы при выполнении основного плана не возникало ни одного вопроса.

Реализация алгоритмов планирования для искусственного интеллекта игровой программы основывается на той же концепции, что и ранее. Имеется объект, управляемый системой ИИ, и необходимо, чтобы этот объект следовал некоторому плану для достижения некоторой цели. Следовательно, необходимо смоделировать этот план с помощью некоторого языка программирования; обычно это C/C++, но можно использовать и некоторый специализированный высокоуровневый язык сценариев. В любом случае, кроме моделирования плана, нужно смоделировать все объекты, являющиеся частью плана: действия, цели и условия. Каждый из перечисленных элементов может быть представлен простой структурой или классом C/C++. Например, цель может выглядеть следующим образом:

```
typedef struct GOAL_TYP
{
int class; // Класс цели
char* name; // Название цели
int time; // Время, отпущенное на достижение цели
int* subgoals; // Указатель на список подцелей,
// которые должны быть достигнуты
int (*eval)(void); // Указатель на функцию, опреде-
ляющую,
// достигнута ли цель
// Прочие данные
} GOAL *GOAL_PTR;
```

Элегантный метод кодирования плана состоит в использовании правил продукции, или просто продукций, и деревьев принятия решений (рис. 70). Продукция представляет собой логическое утверждение с рядом посылок и следствием:

```
IF X OP Y THEN Z.
```

Здесь *X* и *Y* – посылки, *Z* – следствие, а *OP* может быть любой логической операцией, например, *AND*, *OR* и т.д.

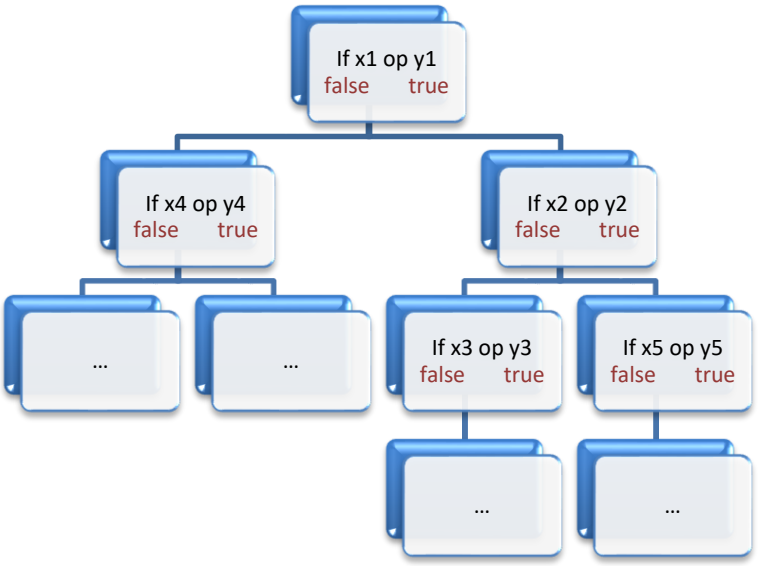


Рис. 70. Дерево принятия решений

Фактически план является не чем иным, как набором условных операторов вместе с действиями и целями

Пример. План тактики ведения огня:

- Если игрок близко и повреждения малы, то атаковать игрока.
- Если игрок далеко и горючего много, то искать игрока.
- Если повреждения велики и игрок близко, то убегать от игрока.
- Если повреждения велики, боеприпасов нет и игрок близко, то самоуничтожиться.

Поиск пути (pathfinding) представляет собой вычисление и выполнение движения по пути из точки *p1* к целевой точке *p2*. При отсутствии препятствий простейший способ попасть в целевую точку – двигаться по прямой в направлении цели, пока целевая точка не будет достигнута. Однако задача существенно усложняется, если на пути встречаются препятствия, которые придется огибать.

Когда на пути появляются небольшие выпуклые препятствия, обычно используется алгоритм, состоящий в том, чтобы при столкновении с препятствием отойти назад, выполнить поворот вправо или влево на 45 – 90° и переместиться на некоторое предопределенное расстояние (*AVOIDANCE_DISTANCE*). После этого искусственный интеллект вновь определяет направление на целевую точку и повторяет попытку добраться к ней по прямой (рис. 80).

Другой метод обхода препятствий состоит в обходе по контуру. Данный алгоритм отслеживает контур препятствия на дороге объекта. Реализация данного алгоритма состоит в перемещении объекта вокруг препятствия и в периодической проверке, продолжает ли отрезок между объектом и целью пересекать препятствие. Если отрезок не пересекает препятствие, то объект может двигаться по направлению к своей цели; в противном случае обход препятствия продолжается.

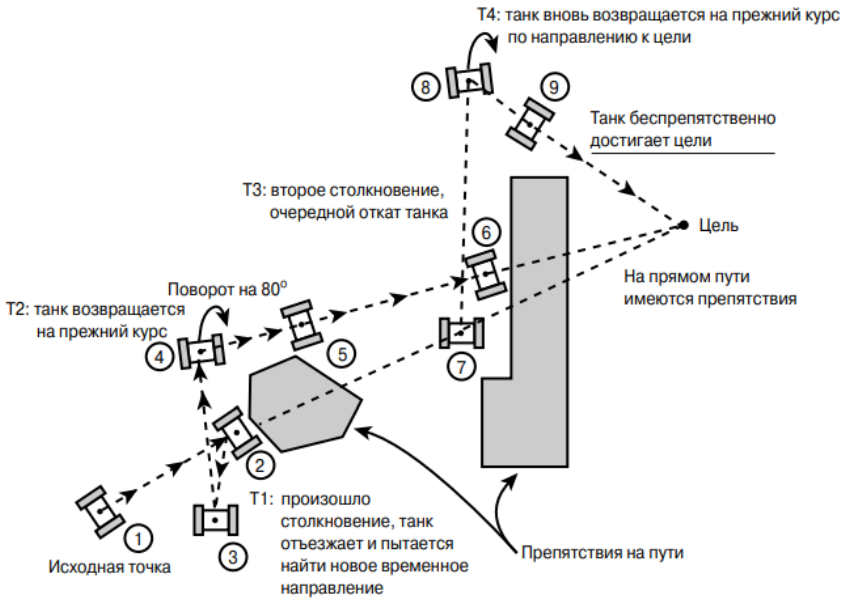


Рис. 80. Метод проб и ошибок при поиске пути

Этот алгоритм вполне работоспособен, хотя и выглядит не самым «умным», поскольку, как правило, объект движется не по очевидно кратчайшему пути. Можно комбинировать два описанных метода поиска пути: сначала объект движется методом проб и ошибок и если не достигает цели за заранее заданное время, то в действие вступает алгоритм обхода по контуру.

Алгоритмы надежного поиска пути:

- Поиск в ширину. Этот поиск одновременно охватывает все направления. Пользователь посещает прежде всего все узлы, доступные из стартовой точки, затем все узлы, доступные из уже посещенных, и т.д. Это достаточно простой и неэффективный алгоритм, поскольку он никак не учитывает интересующее нас направление на цель. Вот как выглядит псевдокод этого алгоритма (рис. 90).
- Двунаправленный поиск. Этот алгоритм аналогичен только что рассмотренному с тем отличием, что одновременно выполняются два поиска: один – из исходной точки, а второй – со стороны цели. Когда они перекрываются, вычисляется кратчайший путь.

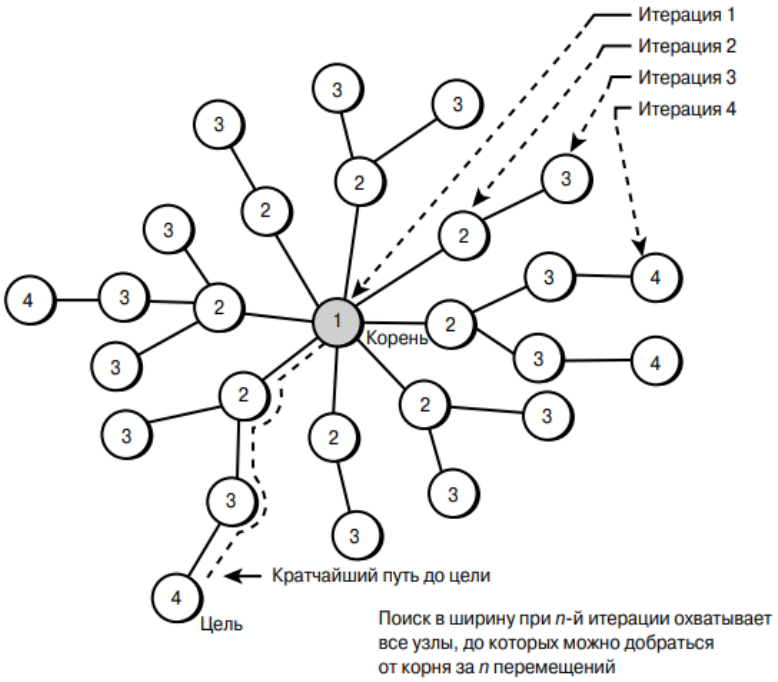


Рис. 90. Поиск пути в ширину

- Поиск в глубину. Этот вид поиска противоположен поиску в ширину. Поиск в глубину просматривает одно направление до тех пор, пока узлы не исчерпаются или пока не будет достигнута цель. Затем выполняется поиск в другом направлении и т.д. (рис. 91).

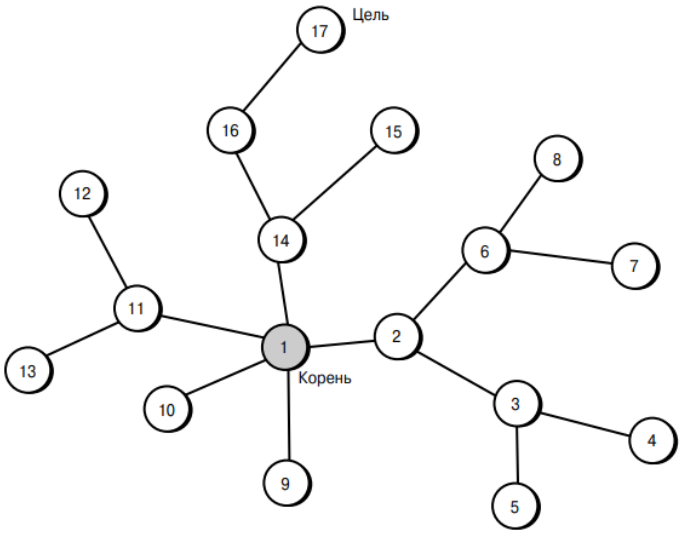


Рис. 91. Поиск в глубину

- Поиск Дейкстры. При каждой итерации алгоритм Дейкстры определяет кратчайший путь к очередной точке и затем продолжает работу уже исходя из этой точки (рис. 92).

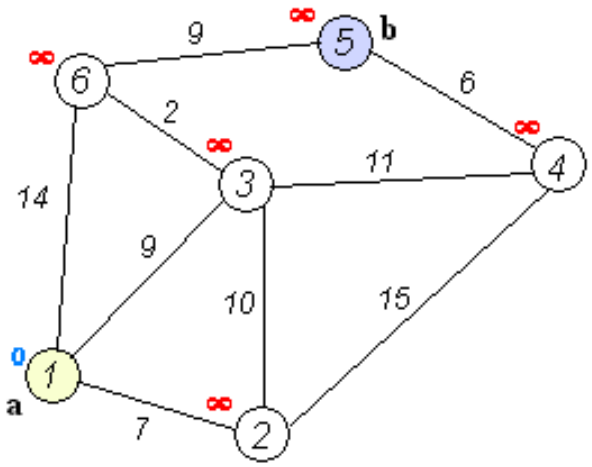


Рис. 92. Поиск Дейкстры

- Алгоритм поиска A*. Порядок обхода вершин определяется эвристической функцией «расстояние + стоимость» (обычно обозначаемой как f(x)). Эта функция – сумма двух других: функции стоимости достижения рассматриваемой вершины (x) из начальной вершины (обычно обозначается как g(x) и может быть как эвристической, так и нет) и функции эвристической оценки расстояния от рассматриваемой вершины к конечной вершине (обозначается как h(x)) (рис. 93).

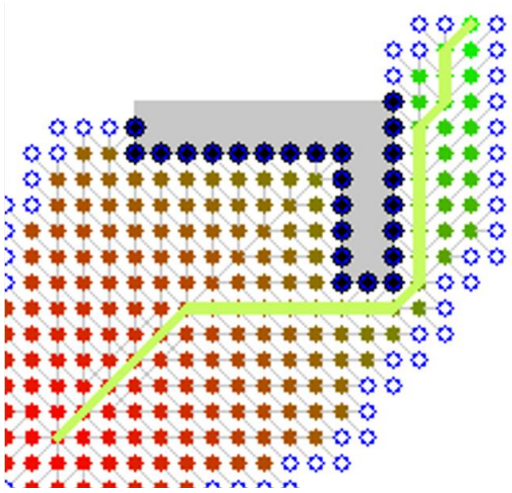
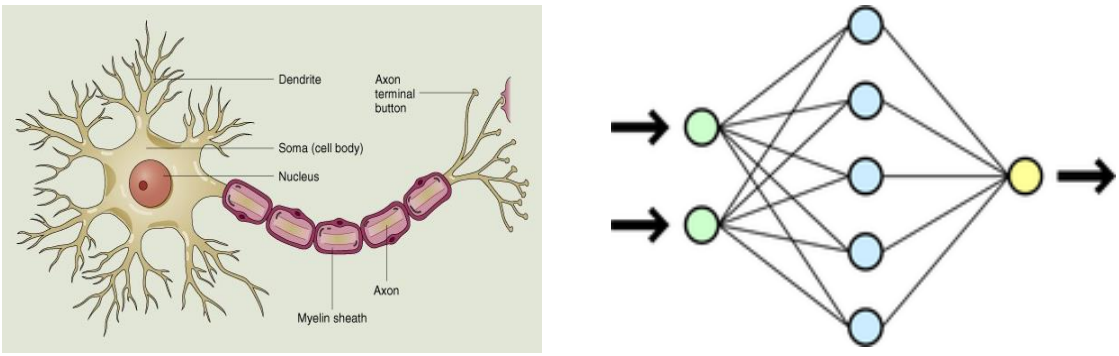


Рис. 93. Алгоритм поиска A*

Искусственные нейронные сети – это одна из тех вещей, о которых все слышали, но никто не видел. Тем не менее в области искусственных нейросетей в последние годы произошли существенные сдвиги. Нейронная сеть представляет собой модель человеческого мозга. Мозг содержит от 10 до 100 млрд клеток, каждая из которых может обрабатывать и пересылать информацию.

Основными частями нейрона являются сома, аксон и дендриты. Сома представляет собой основное тело клетки и выполняет обработку сигналов, в то время как аксон передает сигналы дендритам, доставляющим их другим нейронам.

Каждый нейрон выполняет очень простую функцию – обработать входной сигнал и послать или не послать выходной сигнал. Нейроны имеют множество входов, единый выход (который может быть разделен). Существует некоторое правило, в соответствии с которым обрабатываются входные и генерируются выходные сигналы нейрона (рис. 94). Правила обработки сигнала чрезвычайно сложны; достаточно лишь сказать, что входные сигналы некоторым образом суммируются и полученный результат приводит к передаче выходного сигнала.



а) б)

Рис. 94. Строение живого нейрона (а); искусственная нейросеть (б)

Применение нейросетей при обучении ИИ имеет широкое применение – от анализа транспортной ситуации на дорогах [6] до создания фотореалистичной графики в виртуальной реальности [18].

Вот краткий экскурс в практику ИИ в системах виртуальной реальности. Подытожим все сказанное.

Если в системе виртуальной реальности применяются объекты с простым поведением, такие как камни, ракеты и т.п., то для них используют простые детерминированные алгоритмы.

Для объектов, которые предполагаются «разумными», но являются скорее частью окружающей среды, чем действующими объектами (например, летающие вокруг птицы, пролетающий рядом космический корабль), используют детерминированный ИИ с элементами случайного выбора.

Для важных персонажей систем виртуальной реальности, взаимодействующих с игроками, безусловно, необходимо использовать конечные автоматы вместе с другими вспомогательными методами. Однако некоторые персонажи могут быть «не очень умными». В таком случае не следует затрачивать излишние усилия на их персонализацию посредством специализированных распределений вероятностей или терять время на разработку самообучающихся систем.

При реализации «очень умного» персонажа игры нужно собрать все вместе: и детерминированные алгоритмы, алгоритмы работы с состояниями, алгоритмы обучения. У такого персонажа ИИ должен быть конечным автоматом с применением большого количества условной логики, вероятностей, с запоминанием переходов между состояниями.

1.8. Программно-аппаратная реализация запахов, тактильных ощущений

Для управления виртуальной реальностью необходимы специальные устройства, обеспечивающие обратную связь. Устройства для физического погружения в виртуальную реальность (англ. Devices for a Physical Immersion into Virtual Reality) – это технические решения взаимодействия с виртуальным миром, способные задействовать от 2-х чувств человека из основных (зрение, слух, осязание, обоняние, вкус), а также имитировать ходьбу.

Помимо визуального изображения и звука такие устройства предусматривают тактильную связь (англ. haptic feedback – обратная тактильная связь), к которой может относиться положение человека в пространстве, а также симуляция вкуса, симуляция запаха и т.д.

В данной теме рассматриваются различные устройства для управления в ВР-средах, а также поднимается проблема цифрового обоняния.

1.6.1. Манипуляторы для управления в виртуальной реальности

Появление джойстиков датируется началом 60-х XX в и обусловлено появлением первых компьютерных игр. Создание первого джойстика связывают с именами профессора Марвина Мински и студента Массачусетского технологического института Стефена Рассела, больших поклонников компьютерных игр (рис. 95).

Первые примитивные цифровые джойстики представляли собой стержень, укрепленный на крестовине, имеющей четыре электрических контакта. Чтобы выбрать одно из четырех направлений, нужно наклонить стержень в соответствующую сторону. При замыкании сразу двух контактов добавляются еще четыре направления

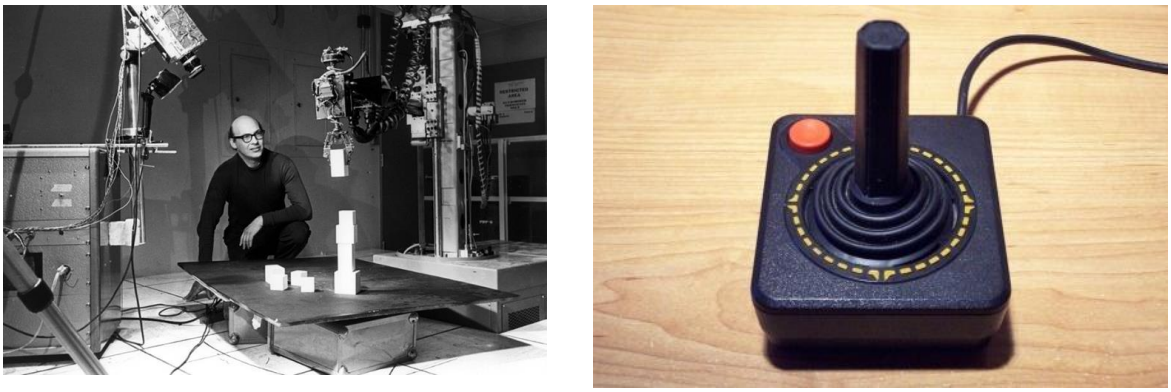


Рис. 95. Первые модели джойстиков

В устройстве аналогового джойстика основание стержня проходит через валик и подвеску, подсоединенные к потенциометрам. Каждый потенциометр регистрирует движение в своей плоскости. Потенциометр (от лат. potentia – сила и метр) – электроизмерительный компенсатор, прибор для определения ЭДС или напряжений компенсационным методом измерений. С использованием мер сопротивления потенциометр может применяться для измерения тока, мощности и др. электрических величин, а с использованием соответствующих измерительных преобразователей – для измерения различных неэлектрических величин (например, температуры, давления, состава газов). Различают потенциометры постоянного и переменного тока [39].

Рис. 96. Современный джойстик

Со временем джойстики снова стали цифровыми, но уже другого уровня. Такой джойстик генерирует сигнал обычно на базе тех же технологий, что и аналоговый, а затем оцифровывает этот сигнал и передает его в компьютер. Преимущество этого решения заключается в том, что аналоговый сигнал превращается в цифровой до того, как он попадает в игровой порт (сильно зашумленное в электронном смысле пространство) (рис. 96).

Потенциометры имеют скользящие контакты, которые засоряются пылью и продуктами окисления, что ухудшает контакт и может приводить к проблемам управления. Этих недостатков лишены оптические джойстики, в которых вместо резисторов используются оптические сенсоры, менее подверженные износу [39].

У устройств управления есть такая характеристика, как степень свободы. Степень свободы – характеристика движения механической системы. Число степеней свободы определяет минимальное количество независимых переменных (обобщённых координат), необходимых для полного описания движения механической системы (рис. 97).

Рис. 97. Степени свободы джойстика

Наше пространство трехмерное. Трем осям соответствует максимальное количество степеней свободы, равное шести: три смещения вдоль осей и три поворота вокруг этих осей [9].

Троттл – это контроллер тяги, еще одно устройство управления в ВР. Чаще всего применяется в симуляторах летательных аппаратов. Направление движения – вперед-назад (рис. 98).

Рис. 98. Троттл

Маркировка осей джойстика и троттла:

- X1 (или X) – наклон рукоятки вперед/назад (тангаж);
- Y1 (или Y) – наклон рукоятки вправо/влево (крен);
- X2 (или rZ) – педали или поворот рукоятки вокруг своей оси (хвостовой руль);
- Y2 (или Z) – вперед/назад тротл (тяга).

Компьютерный руль – это игровой контроллер, имитирующий автомобильный руль (рис. 99). Обычно функции управления присваиваются осям плавного регулирования следующим образом:

X1 – руль вправо/влево;

Y1 – газ/тормоз или X1 – руль вправо/влево;

Y1 – газ;

X2 – тормоз.

Рис. 99. Руль

Одним из основных параметров является угол поворота руля: чем он больше, тем лучше. Угол поворота руля колеблется от 100 до 270°. Руль является наследником джойстика: первые рули действительно эмулировали двухосный джойстик.

Геймпад, или, как его еще иногда называют, игровой планшет – это манипулятор для управления обеими руками. Левой рукой обычно контролируется движение, а правой – разнообразные действия посредством нажатия кнопок (рис. 100).

Рис. 100. Геймпад

Внутри геймпада установлены двигатели, которые управляются игрой и в определенные моменты заставляют его вибрировать. Во-первых, это добавляет ощущений: кроме визуального и звукового воздействия, игрок подвергается еще и тактильному. Во-вторых, сам игровой процесс становится азартнее, хотя и несколько сложнее – управлять вырывающимся из рук манипулятором довольно забавно. Единственное «но» – врачи категорически не рекомендуют геймпады с обратной связью детям дошкольного и младшего школьного возраста.

Если хочется добиться максимального эффекта присутствия в VR, лучше использовать VR-перчатки (рис. 101). Они позволяют отслеживать движения не только рук, но даже пальцев, могут обладать обратной тактильной связью и существенно расширяют список доступных действий в играх виртуальной реальности. Каждая перчатка оснащается набором датчиков (акселерометрами, магнитометрами, гироскопами, барометрами, сенсорами) и способна распознавать движения рук и пальцев, передавая информацию в VR-приложение [43].

Рис. 101. Сенсорная перчатка

Перчатки для шлемов виртуальной реальности представляют собой довольно сложное устройство. Гаджет оснащен многочисленными датчиками, с помощью которых происходит отслеживание движений руки и пальцев на ней. Наиболее продвинутые модели способны регистрировать сгибы пальцев в фалангах. На запястье обычно располагается микропроцессор, модуль беспроводной связи и аккумулятор. Впрочем, батареи может и не быть, зачастую перчатки подключаются компьютеру или консоли через кабель.

Несмотря на сложное техническое исполнение устройства, принцип работы довольно прост. Как только пользователь изменяет положение руки/пальца/пальцев или совершает рукой определенные действия/жесты, информация о новых координатах передается в приложение на ПК.

Картинка в шлеме динамически изменяется: пользователь видит виртуальную проекцию собственных рук, повторяющих движения в реальном мире. Если пользователь совершает некие действия, к примеру воспроизводит в реальности открытие двери рукой, в VR-приложении открывается виртуальная дверь.

Отслеживание перемещения рук и пальцев пользователя, а также замер скорости перемещений производятся с помощью гироскопов, магнитометров и акселерометров. Присутствие в устройстве барометра позволяет дополнительно оценивать степень давления, создаваемого руками или пальцами игрока.

В современных моделях все датчики спрятаны внутри перчаток, именно поэтому по внешнему виду они мало чем отличаются от обычных перчаток. Первые устройства нередко оснащались инфракрасными сенсорами, располагаемыми на каждом пальце. Специальная инфракрасная камера получала данные о перемещении сенсоров. В новых устройствах внешних датчиков нет, равно как нет и камер. Модели для геймеров все чаще оснащаются небольшими вибромоторами для обратной тактильной связи.

У сенсорной перчатки 6 степеней свободы (рис. 102).

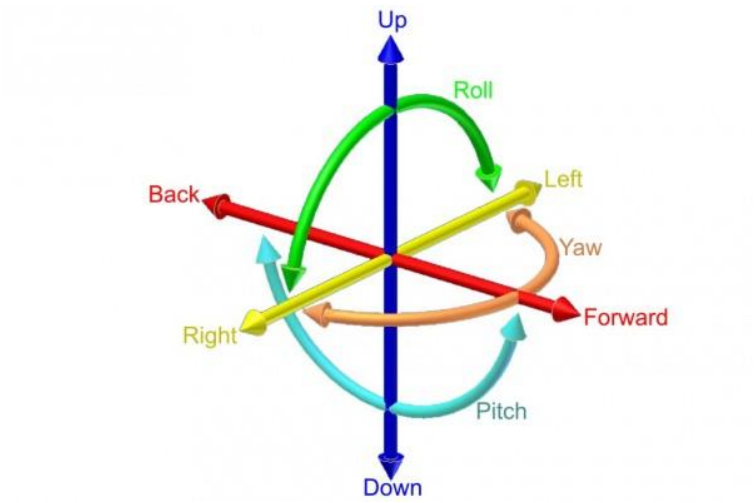


Рис. 102. Распределение шести степеней свободы в пространстве

3D-картинку в настоящих VR-системах получают с помощью таких устройств, как VR-шлем или VR-очки. VR-шлемы оснащены двумя мониторами (или одним, который разделён на две части). Каждый из этих мониторов показывает отдельные изображения для каждого глаза. А чтобы правильно сфокусировать взгляд, используются линзы.

В сочетании с особым устройством трекингом, очки виртуальной реальности также становятся устройством управления. Трекинг – одна из технологий ВР, лежащая в основе взаимодействия человека с виртуальным миром. Предназначена для определения позиции и ориентации реального объекта (например, руки, головы или специального устройства) в виртуальной среде с помощью нескольких степеней свободы. Как правило, трёх координат расположения объекта (x, y, z) и трёх углов, задающих ориентацию объекта в пространстве («крен», «тангаж», «рыскание» или углы Эйлера).

Определение позиции и ориентации реального объекта в пространстве определяется при помощи специальных датчиков и маркеров. Датчики снимают сигнал с реального объекта при его перемещении и передают полученную информацию в компьютер. В VR-шлемах трекер отслеживает движения головы, и соответственно изменяется изображение в шлеме. Для людей с ограниченной подвижностью трекер применяется как замена мыши.

В симуляторах применяется для имитации поворотов головы. Особенно это важно в авиасимуляторах, где вражеский самолёт может находиться в любом направлении от вас. Для того, чтобы пользователь не отворачивался от экрана, система настраивается так, что небольшой поворот головы соответствует повороту камеры на 180°. В трёхмерном программном обеспечении для имитации трёхмерности «заэкранного» мира – при поворотах, наклонах и т.д. изображение на экране подстраивается так, что возникает иллюзия, как будто пользователь смотрит через окно-монитор на трёхмерный объект.

Одним из лидеров устройств для ВР является компания HTC. Популярный комплект Vive Pro доступен школам, технопаркам, кванториумам, университетам (рис. 103).

Рис. 103. Vive Pro 2

По заявлению HTC, экраны шлема Vive использует частоту обновления в 90 Гц. Разрешение экрана составляет 2160x1200. В устройстве используется множество датчиков, в частности: гироскоп MEMS, акселерометр, лазерные датчики позиционирования. Дополнительно с шлемом используются два ручных контроллера. Для точного отслеживания положения HTC Vive и его контроллеров в пространстве (на площади до 4,5 м * 4,5 м) используются две пассивные внешние станции Lighthouse.

Компания Valve выпустила программный набор OpenVR SDK (обновление более раннего Steamworks VR API) с примерами по использованию оборудования SteamVR. В наборе поддерживается версия шлема HTC Vive для разработчиков, контроллер SteamVR и станции позиционирования Lighthouse.

Vive Pro, выпущенный в январе 2018, оснащен дисплеями с более высоким разрешением в 1440x1600 пикселей на глаз, а также второй наружной камерой, присоединяемыми наушниками, микрофоном для шумоподавления. Он имеет разъемы USB 3.0 и DisplayPort для подключения к ПК. Скрытый разъем USB-C можно использовать для подключения дополнительного USB-устройства.

В 2015 г. Epic Games объявила о поддержке HTC Vive в версии движка Unreal Engine 4 для SteamVR. Это дает возможность разработчикам создавать свои виртуальные проекты для широкого круга пользователей, в том числе для бизнеса и образования.

Более полноценные VR-системы включают также и VR-костюм. Он состоит из обтягивающего комбинезона со множеством магнитных сенсоров, которые отслеживают движения всех частей тела. К нему добавляется HMD, датчик(и) кисти (реже перчатка) и провода для присоединения всего этого к компьютеру (рис. 104). Тогда уж точно будет полный комплект ощущений. В основе его принципа действия лежит электротактильная система обратной связи, позволяющая игроку в полной мере ощущать свое присутствие в ВР.

Рис. 104. Костюм виртуальной реальности

Помимо очевидных преимуществ полноты ощущения от виртуальной реальности, костюм имеет ряд минусов: костюм оказывает влияние на здоровье в виде чрезмерных нагрузок на нервные окончания; большие затраты на производство приводят к высокой стоимости продукта, что негативно сказывается на его продаже и целесообразности. В силу дороговизны и громоздкости этого оборудования, оно не нашло широкого применения, даже в профессиональных областях.

Тем не менее костюмы виртуальной реальности связаны с еще одной революционной технологией, которую применяют в киноиндустрии и научных исследованиях. Эта технология называется motion capture (захват движения). В ее основе лежит фиксация сигналов, поступающих с датчиков (маркеров), установленных на живых актерах (примеры таких систем вы видите на снимках, а с результатами их работы регулярно сталкиваетесь на экранах мониторов и телевизоров). Полученная информация о движениях служит основой для реалистичного воссоздания движений компьютерных персонажей. Достоинства технологии очевидны, что касается ее недостатков, то одним из наиболее заметных остается высокая стоимость (рис. 105) [47].

Рис. 105. Технология motion capture при съемке фильма «Железный человек»

Технологии реализации motion capture делятся на 2 вида: маркерные системы и безмаркерные системы.

Маркерная система motion capture, использует специальное оборудование. На человека надевается костюм с датчиками, человек производит движения, требуемые по сценарию, встаёт в условленные позы, имитирует действия. Данные с датчиков фиксируются камерами и поступают в компьютер, где сводятся в единую трёхмерную модель, которая точно воспроизводит движения актёра и на основе которой позже (или в режиме реального времени) создаётся анимация персонажа. Также этим методом воспроизводится мимика актёра (в этом случае на его лице располагаются маркеры, позволяющие фиксировать основные мимические движения) (рис. 106).

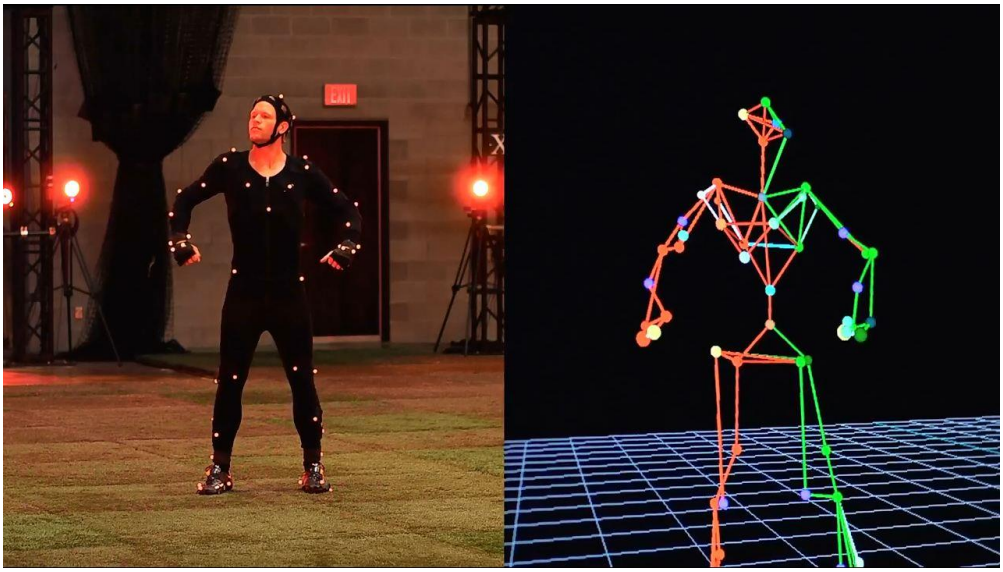


Рис. 106. Маркерная система motion capture

Оптическая пассивная система работает следующим образом. На костюме, входящем в комплект такой системы, прикреплены датчики-маркеры, которые названы пассивными, потому что отражают только посланный на них свет, но сами не светятся. В таких системах свет (инфракрасный)

на маркеры посылается с установленных на камерах высокочастотных стробоскопов и, отразившись от маркеров, попадает обратно в объектив камеры, сообщая тем самым позицию маркера.

Минус оптических пассивных систем заключается в длительности размещения маркеров на актёре. Иногда при быстром движении или близком расположении маркеров друг к другу система может их путать (поскольку эта технология не предусматривает идентификации каждого маркера по отдельности).

Оптические активные системы вместо светоотражающих маркеров, которые крепятся к костюму актёра, используют светодиоды с интегрированными процессорами и радиосинхронизацией. Каждому светодиоду назначается ID (идентификатор), что позволяет системе не путать маркеры друг с другом, а также узнавать их, после того как они были перекрыты и снова появились в поле зрения камер. Во всём остальном принцип работы таких систем схож с пассивными [47] (рис. 107).

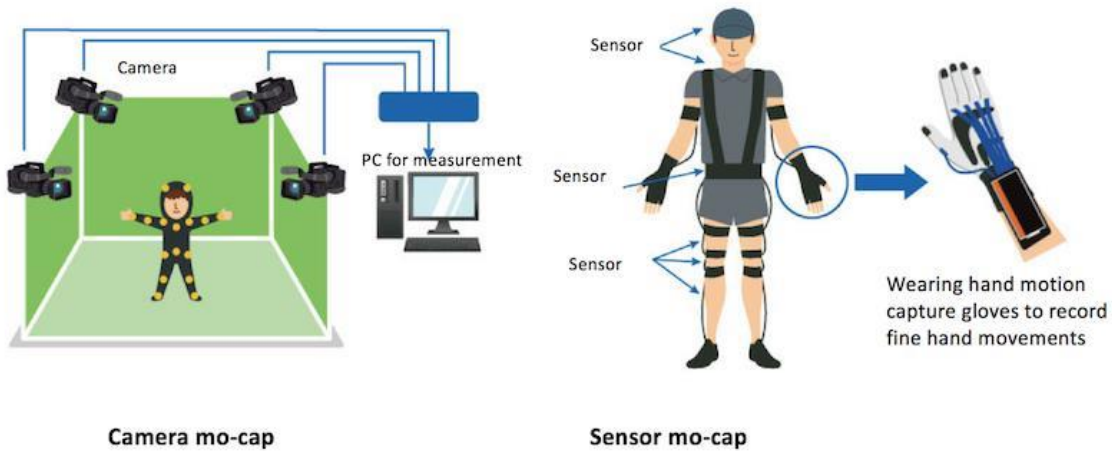


Рис. 107. Безмаркерные системы motion capture

Минусы активных систем следующие: отсутствие возможности захвата движений и мимики лица; дополнительный контроллер, крепящийся к актёру и подключенный к маркерам-светодиодам, сковывает движения; хрупкость и относительно высокая стоимость маркеров-светодиодов.

Механические системы напрямую следят за сгибами суставов. Для этого на актёра надевают специальный механический mocap-скелет, который повторяет все движения актера. В компьютер при этом передаются данные об углах сгибов всех суставов.

Минусы механических систем:

- Мосар-скелет с дополнительным контроллером, прикреплённым к актёру и подключенным к сенсорам сгибов, а в некоторых случаях и провода, тянущиеся от скелета, сильно ограничивают движения актёра.
- Отсутствует возможность захвата движений и мимики лица, движений тесного взаимодействия двух и более актёров (борьба, танцы с поддержками и т.д.), движений на полу (кувырков, падений и т.д.).
- Существует риск поломки механики при неосторожном использовании.

Передача движения в маркерных системах motion capture осуществляется гироскопическими/инертными системами для сбора информации о движении. Эти системы используют миниатюрные гироскопы и инертные сенсоры, расположенные на теле актёра, так же, как и маркеры или магниты используют другие mocap-системы. Данные с гироскопов и сенсоров передаются в компьютер, где происходит их обработка и запись. Система определяет не только положение сенсора, но также угол его наклона.

Минусы гироскопических/инертных систем:

- Отсутствие возможности захвата движений и мимики лица.
- Дополнительный контроллер, прикреплённый к актёру и подключенный к магнитным маркерам, или даже связка проводов, тянущаяся от актёра к компьютеру.
- Высокая стоимость гироскопов и инертных сенсоров.
- Необходимость дополнительной мини-системы (оптической или магнитной).

Безмаркерная технология, не требующая специальных датчиков или специального костюма, основана на технологиях компьютерного зрения и распознавания образов. Актер может сниматься в обычной одежде, что

сильно убыстряет подготовку к съемкам и позволяет снимать сложные движения (борьбу, падение, прыжки, и т.п.) без риска повреждения датчиков или маркеров. Несколько практически применимых безмаркерных систем были разработаны в последние годы, хотя исследования подобной технологии проводятся уже долгое время. На сегодняшний день существует программное обеспечение «настольного» класса для безмаркерного захвата движений. В данном случае не требуется специального оборудования, специального освещения и пространства. Съёмка производится с помощью обычной камеры (или веб-камеры) и персонального компьютера.

1.8.2. Проблемы синтеза запаха. Идеи и разработки в области создания запаха

Запахи – это проблема для науки. Цвет, например, можно легко разложить по спектру. Из 3–4 чистых цветов можно создать практически любой оттенок. Убедительной классификации ароматов до сих пор не существует. Одни ученые делят запахи на «эфирные» (ацетон), «бальзамические» (ваниль), «чесночные» (сероводород) и т.д. Другие выделяют ароматы «фруктовые», «цветочные», «пряные», «горелые». Классификаций запахов множество, но нет ни одной общепризнанной.

С физиологией запахов тоже все непросто. Относительно ясная картина сложилась только в конце XX в. В 1991 г. Ричард Аксел и Линда Бак открыли семейство генов, вырабатывающих протеины, необходимые для улавливания запахов. За это им была вручена Нобелевская премия.

Но самая большая проблема науки – научиться манипулировать запахами. Мы можем без труда сфотографировать розу и записать картинку на компьютер. С оцифрованным изображением цветка мы сумеем сделать все что угодно – менять оттенки, увеличивать яркость, изменять пропорции. Несколькими движениями мыши можно отправить эту фотографию от вас в

какую угодно точку на Земле. Но аромат цветка остается неподвластным цифровым технологиям.

В отличие от технологии передачи звука или видео, когда соответственно звук или изображение вначале фиксируется прибором, а затем превращается в цифровой код, в индустрии запахов такой технологии пока не существует. Нет прибора, подобного кинокамере, позволяющего записывать сам запах или его изменение во времени. По-прежнему основным инструментом при «записи» запаха является человеческий нос. Занимаются анализом запаха специалисты, которые раскладывают каждый новый запах на составляющие и соответственно подбирают количество компонентов, необходимых для воссоздания этого запаха, после чего аромат записывается в виде небольшого файла размером около двух килобайт.

Синтезатор запаха Pinoke имеет в верхней части «гребешок». В нём располагается какой-то минимальный набор (запас) химических веществ, впоследствии преобразуемых в запах. Устройство имеет некий набор химических реактивов и из них синтезируется запах. Устройство напоминает принтер, только вместо цветов CMYK в картридже химические вещества. Однако подобный прибор не был представлен разработчиками для тестирования (рис. 108).

Рис. 108. Синтезатор запаха Pinoke

Молодая американская компания DigiScents была одной из первых, кто заявил о создании устройства электронного запаха – iSmell. Оно имеет сменный картридж, содержащий более 100 различных ароматических веществ, которые по команде компьютера смешиваются, образуя необходимый запах. iSmell или iSmell Personal Scent Synthesizer подключался к ПК и начинал пахнуть, как только пользователь посещал сайт или открывал электронное письмо со встроенным кодом активации. Устройство содержало картридж со 128 основами ароматов, которые можно было смешивать для создания различных запахов [49].

На сегодняшний день фирма свернула разработки в этой области и в промышленное производство прибор так и не попал.

Синтез запахов происходит в приборе iSmell. Этот прибор (в основе которого находится картридж с элементарными составляющими запахов) подключается к PC через порт LPT или USB. Соответственно приняв код того или иного запаха, компьютер передает команду в iSmell устройству управления подачей составляющих картриджа, в результате чего происходит «впрыскивание» необходимых компонентов. Перемешивание составляющих интенсифицируется действием подогревающей спирали и подается наружу под действием обычного вентилятора. Интенсивность запаха определяется специальным регулятором (рис. 109).

После того как запах будет представлен в виде цифрового кода, он соответственно может быть передан традиционным способом. Причем малые размеры файла не накладывают ограничений на пропускную способность канала и при передаче вместе с изображением практически не увеличивают объем передаваемых данных. Обычно файл, передающий запах, связан с тем или иным изображением. Так, например, если на веб-странице вы щелкаете мышью на том или ином названии цветка, то вместе с его изображением компьютер начнет выдавать соответствующий запах.

Рис. 109. Прибор синтеза запаха iSmell

Ближе всего к созданию промышленных образцов манипулятора запахами подошли в Японии. Как сообщают информационные агентства, группа ученых из Национального университета Сингапура (NUS) и Университета Кэйо (Япония) с 2013 г. ведет работу над новой методикой, позволяющей оцифровать вкус и цвет. Перед испытуемым ставили прозрачный стакан с дистиллированной водой. В стакане активировали светодиодную подсветку, повторяющую цвет лимонада. Встроенные в верхнюю часть стакана электроды раздражали вкусовые рецепторы языка, имитируя у дегустатора такие же вкусовые ощущения, как при питье лимонада.

Прототип устройства мог воспроизводить лишь четыре базовых вкуса: сладкий, горький, соленый и кислый (поэтому опыт ставили на лимонаде). Ученые планируют добавить к вкусовым ощущениям запахи, чтобы сделать продукты более реалистичными.

Похожую технологию в апреле 2010 г. опробовали в нескольких японских кинотеатрах. Местная телекоммуникационная компания NTT

Communications установила в залах специальное компьютерное оборудование, благодаря которому ключевые сцены фильмов сопровождались соответствующими ароматами. Управлялась система через Интернет.

Ученым пока не удается передать запах и вкус. Возможно, при разработке мозговых интерфейсов эту проблему удастся решить.

1.9. Среда визуального программирования Alice

Alice – это инновационная программная 3D-среда, позволяющая легко создавать анимацию для описания историй, игр или делать видео для публикации в сети. Alice является свободно доступной программой и может являться средством для знакомства студентов с объектно-ориентированным программированием.

Эту программу разработали в университете Карнеги-Меллона в США, и около 10 % американских колледжей используют это ПО для учебных целей. Программа представлена в версиях под Windows, Linux и Mac OS. Также поддерживаются редакции для студентов и для школьников.

Официальный сайт Alice http://www.alice.org/index.php [2].

Alice является свободным и открытым объектно-ориентированным языком программирования для обучения с интегрированной средой разработки (IDE). Он реализован в Java. Alice использует методы drag-and-drop для создания компьютерной анимации с использованием 3D-моделей. Программное обеспечение разрабатывается исследователями в университете Карнеги-Меллона, участвовал в том числе и Рэнди Пауш. Язык был разработан для решения трёх основных задач в образовательных программах.

- В большинство промышленных языков программирования вносится дополнительная сложность. Язык Alice предназначен исключительно для обучения программированию. Он может быть использован при работе с 3D-интерфейсом пользователя. У пользователя есть возможность программировать при помощи стрелок и других элементов, называемых контролами.

• Происходит объединение с IDE. Нет необходимости запоминать синтаксис. Тем не менее Alice полностью поддерживает объектно-ориентированное программирование, событийно-ориентированное программирование.

• Осуществляется направленность на конкретный слой населения, которое, как правило, не использует компьютерное программирование, т.е. представляет простых пользователей.

Один из вариантов языка Alice 2.0 называется «История, рассказанная Алисой» (Алиса – персонаж знаменитого произведения Л. Кэрролла). Этот вариант языка Alice был создан Кетлин Келлер (англ. Caitlin Kelleher) для её докторской диссертации. Версия языка включает в себя 3 основных различия:

- Высокоуровневая анимация. Позволяет пользователям программировать социальные взаимодействия между персонажами.
- Учебник на основе рассказа. Знакомит пользователей с программированием через создание сюжета.
- Галерея 3D-персонажей и декорации с пользовательской анимацией. Позволяет «оживлять» идеи истории.

Работа с Alice позволяет узнать студентам основные понятия программирования в контексте создания анимационных фильмов и простых игр. В Alice 3D-объекты, которые представляют людей, животных, транспортные средства, находятся в виртуальном мире, а студенты создают программу для анимации объектов (рис. 110).

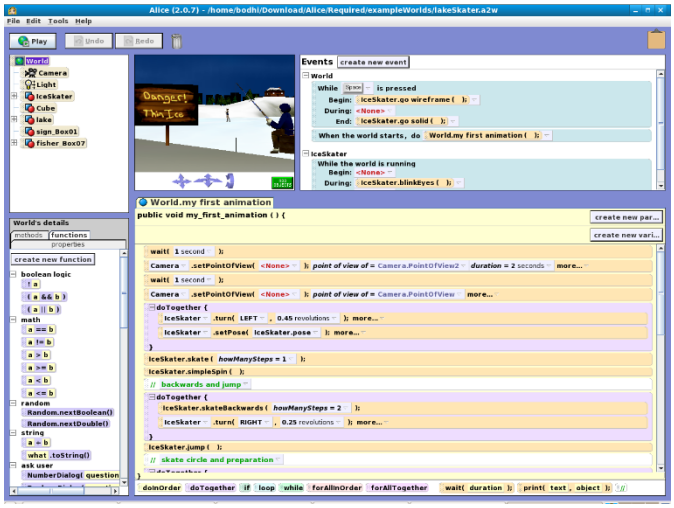


Рис. 110. Интерфейс Alice 2

Программы Alice состоят из алгоритмических конструкций (следования, ветвления, циклов, процедур, функций). Можно создавать сложные конструкции параллельного программирования. Для этого достаточно перетащить в окружение элемент «Do together».

Рабочая плоскость Alice разбита на несколько окон: крайнее левое окно – это дерево объектов, в следующем окне отображается собственно виртуальный мир, имеется также специальная область для событий. Нажатие на кнопку Play запускает программу на выполнение.

Первоначально, в Alice нужно выбрать виртуальный мир, в котором будут находиться персонажи. К ним относятся трава, море, луна, снег, болото, пустыня. Затем выбранный мир нужно населить персонажами. Выбор действительно велик:

- двуногие персонажи (человек, инопланетянин, зайчик, чеширский кот, шляпник и др.);
- птицы;
- водоплавающие (рыбы, водоплавающие млекопитающие);
- четвероногие персонажи (кошка, инопланетный робот, крокодил, корова и др.);
- предметы (деревья, мебель, камни, здания, элементы интерьера и др.).

У каждого объекта существуют ряд свойств и событий. Общие для всех объектов свойства:

- расположение в пространстве,
- размеры,
- прозрачность.

Для двуногих существует большое количество свойств, отвечающих за расположение костей скелета в пространстве: голова, руки, ноги, глаза, талия, шея и др. Для четвероногих персонажей также можно использовать управление различными деталями объектов. Для управления применяются визуальные элементы и программный код. Визуализация управления происходит путем выбора всего персонажа целиком и путем его последующего

перемещения в пространстве, а также выбором определенной части тела и ее последующим изменением.

Управлять объектами можно не только с помощью мыши, но и с помощью меню. Alice обеспечивает много встроенных инструкций, которые могут использоваться, чтобы приспособить размер и положение объекта в сцене. Эти инструкции называются методами. Для иллюстрации использования методов, будем работать с новым миром, где методы будут использоваться для настройки сцены.

Алгоритмические конструкции в языке программирования Alice:

- **Do in Order** – выполнение действий по порядку.
- **Count –** выполнение действий определенное количество раз (аналог цикла с параметром).
- **While** – выполнение действий, пока условие истинно или ложно.
- **For each in** – для каждого объекта из серии.
- **If** – выполнение действия, если условие истинно.
- **Do together** – выполнение совместных действий несколькими объектами (параллельное выполнение программы).
- **Each in together** – каждый из объектов, объединенных.

В следующей главе будут представлены лабораторные работы по этой среде программирования, в которой можно будет создать интересные истории, игры и обучающие программы.

ГЛАВА 2. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

2.1. Лабораторная работа 1. Знакомство со средой Alice

Alice – это инновационная среда программирования, работать в которой можно на основе блоков, а объектами являются 3D-персонажи. В отличие от многих приложений для программирования, Alice мотивирует обучение через творческое исследование. Alice предназначена для обучения навыкам логического и вычислительного мышления, фундаментальным принципам программирования и для первого знакомства с объектноориентированным программированием. Система разработана и поддерживается Carnegie Mellon University, а предлагаемое в ее рамках ПО бесплатно и доступно для всех желающих. Оно существует в версиях для Windows, Mac OS и Linux, а также в двух редакциях – основной, предназначенной для вузов (Alice 3), и упрощенной, нацеленной на применение в школах (Alice 2) [2].

В данном учебном пособии акцент будет сделан на Alice 2 (версия 4) для обучения школьников. Эта версия включает в себя следующие особенности:

- высокоуровневую анимацию, которая позволяет пользователям программировать взаимодействия между персонажами;
- учебник, который знакомит пользователей с основами программирования через создание сюжета;
- галерею 3D-персонажей и декорации с пользовательской анимацией для «оживления» идей истории.

Рассмотрим интерфейс программы (рис. 111):

1. Дерево объектов. Здесь находятся все объекты и подобъекты, которые присутствуют на сцене.
2. Окно цены, в котором находятся все выбранные 3D-объекты.
3. Свойства, методы и функции для управления объектами.

4. Окно кода, в которое можно перетаскивать свойства, методы или функции и в котором можно программировать разные истории.

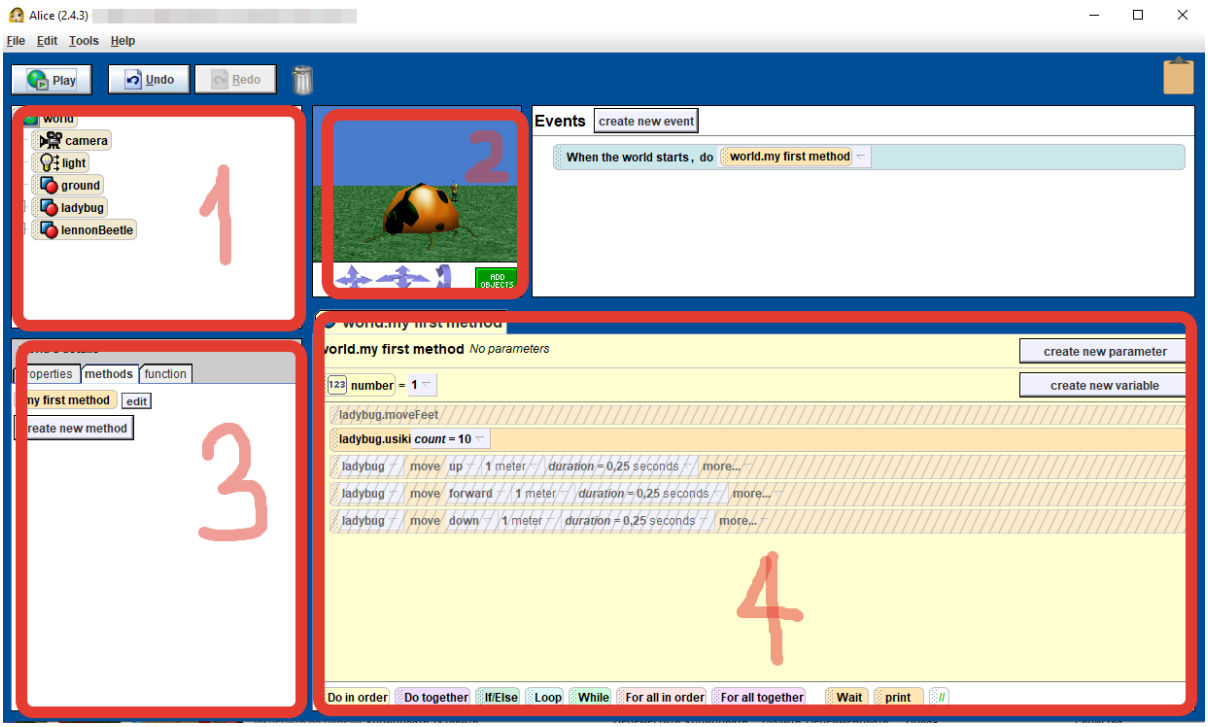


Рис. 111. Внешний вид главного окна

Кнопка Play запускает созданный виртуальный мир на исполнение. При этом открывается отдельное окно (рис. 112).

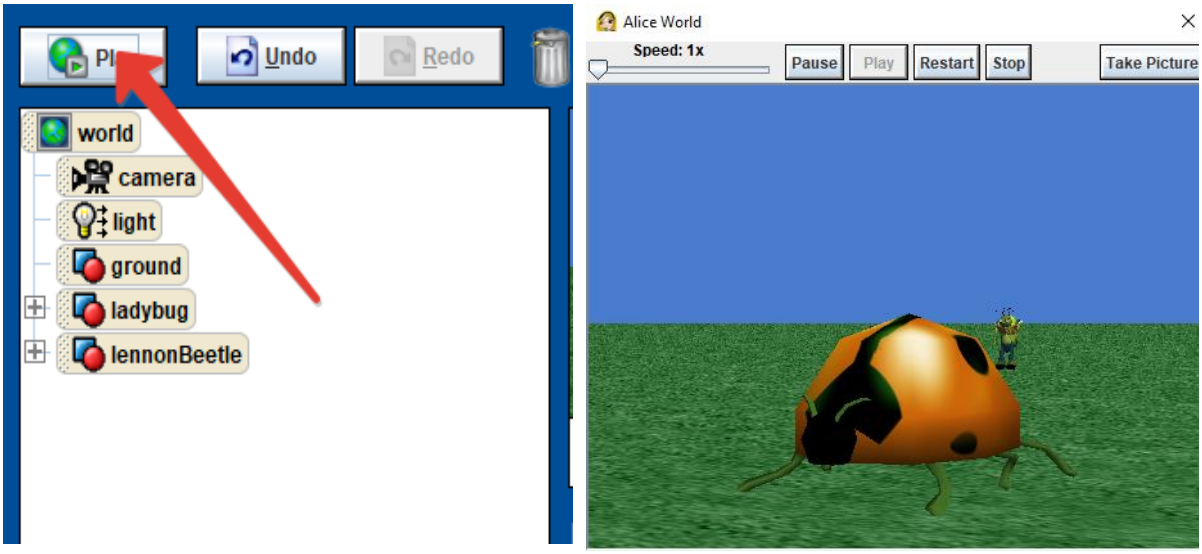


Рис. 112. Запуск программы

В мир Alice объекты добавляются по нажатию кнопки Add objects 

. Примеры объектов представлены на рис. 113.

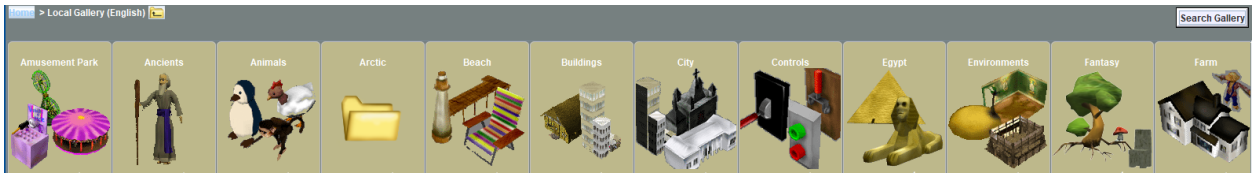


Рис. 113. Примеры объектов Alice

Каждый объект в Alice инкапсулирует свои данные (атрибуты типа Private, такие как высота, ширина и положение в мире) и содержит свои методы. В Alice объекты добавляются крайне просто. К объектам можно добавить методы и свойства, которые принимают необходимые параметры и вызываются при определенных условиях (рис. 114).

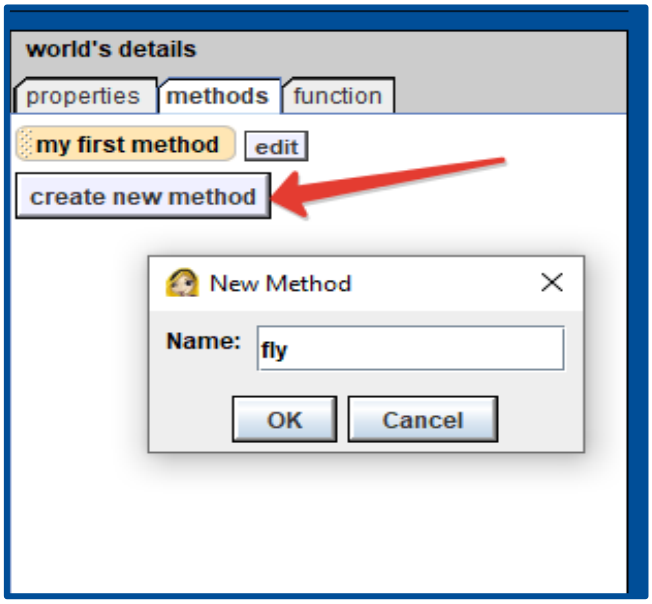


Рис. 114. Добавление нового метода

Программа Alice может содержать переменные, методы, функции, циклы, условные операторы. Всё это можно запрограммировать, используя метод drag-n-drop (простое перетаскивание мышью) (рис. 115).

Рис. 115. Создание кода программы с помощью перетаскивания мышью нужных конструкций

Нужно лишь перетаскивать в окружение программные элементы, такие как условные ветвления, циклы Do...While и Loop (цикл с параметром, петля), а также операторы ожидания Wait, вывода текста print и комментарии. Можно даже задавать сложные конструкции параллельного программирования, перетаскивая в окружение элемент Do Together (рис. 116).

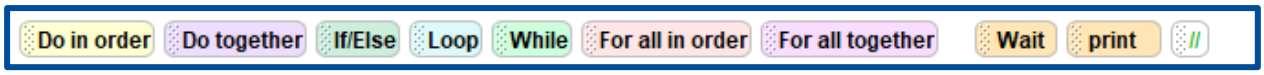


Рис. 116. Алгоритмические конструкции в Alice

- **Do in order** (делай по порядку) – в этом блоке инструкции выполняются последовательно одна за другой.
- **Do together** (делай вместе) – в этом блоке инструкции выполняются параллельно, все одновременно.
- **If/Else** (условная конструкция) – если что-то истина, то делай определённые действия, иначе делай другие действия.
- **Loop** (петля) – цикл многократно выполняет определённый набор действий.
- **While** (цикл с условием) – выполняет действия, пока условие истинно; если условие ложно, прекращает выполнение цикла.

• **For all in order** (цикл для всех по порядку) – выполняет для всех элементов списка определённые действия по порядку.

• **For all together** (для всех вместе) – выполняет цикл для всех объектов списка при этом выполнение происходит одновременно.

• **Wait** (ожидание) – реализует задержку выполнения алгоритма в определенное количество времени.

• **Print** (печать) – распечатывает результат в окне справа.

• **//** (комментарии) – позволяет создавать комментарий, чтобы не заблудиться в программе.

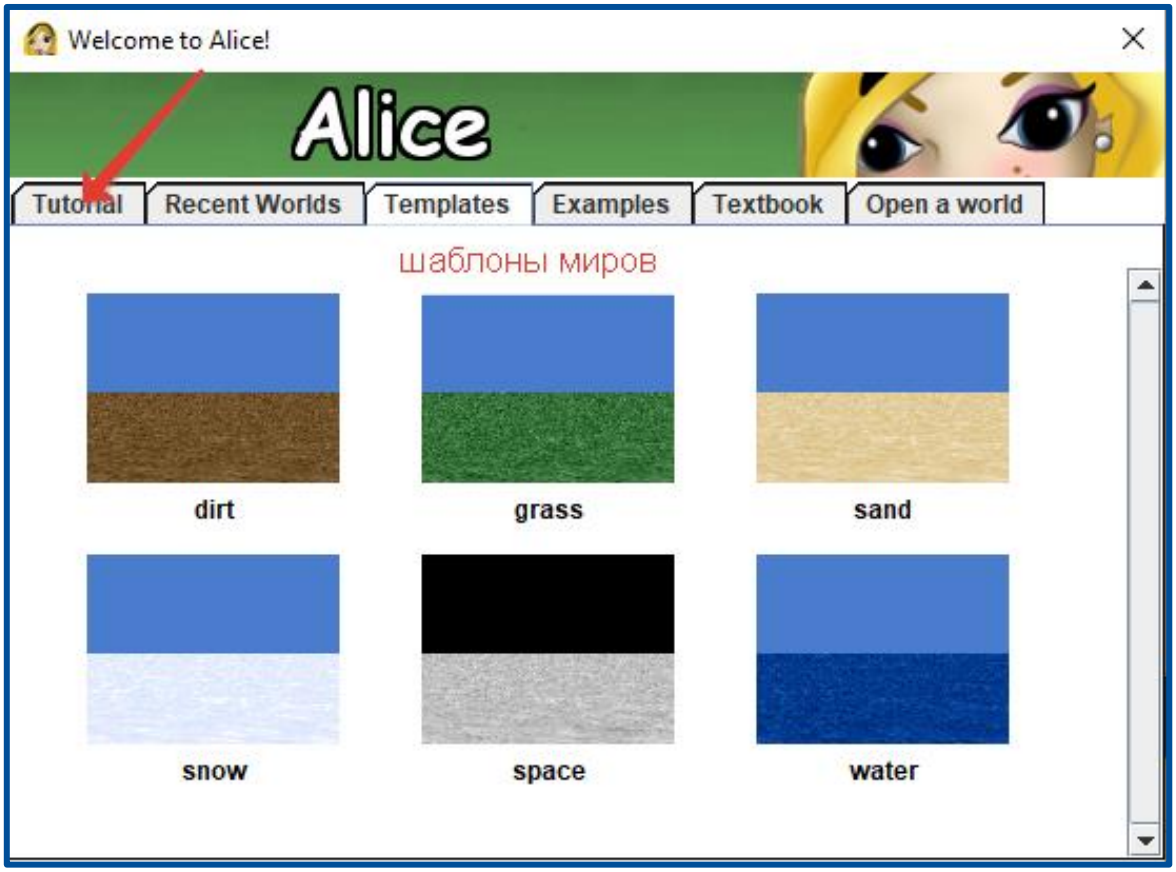


Рис. 117. Стартовое окно в Alice

Задание 1. Запуск Alice и изучение учебного проекта

Запустите программу alice.exe. По умолчанию Alice при запуске показывает окно, как на рис. 117. Это шаблоны миров. Выберите вкладку **Tutorial** для просмотра обучающих примеров.

В качестве обучающего примера выберите проект Tutorial1 (рис. 118). В пошаговом режиме рассмотрите все элементы окна. Запустите проект нажав на клавишу Play.

Рис. 118. Запуск учебного примера

Задание 2. Создание первого проекта

Создадим собственный проект. В нашем проекте главным героем будет зайчик (Bunny). По традиции изучение любого языка программирования начинается с фразы «Hello, world!» («Привет, мир!»). С этой фразы начнём и мы свой проект.

1. Для начала нужно выбрать фон для анимации. Зайдите в меню **File** (файл) и нажмите на пункт меню **New World** (новый мир), у вас появится диалоговое окно, которое мы уже видели ранее. Однако теперь нам нужна

вкладка **Template** (шаблоны) миров. Выберите любой понравившийся вам шаблон мира, например **Grass** (трава).

2. С помощью кнопки **Add Objects** (добавить объекты) добавьте на сцену зайчика (Bunny), он находится в папке **Animals** (Животные) (рис. 119).

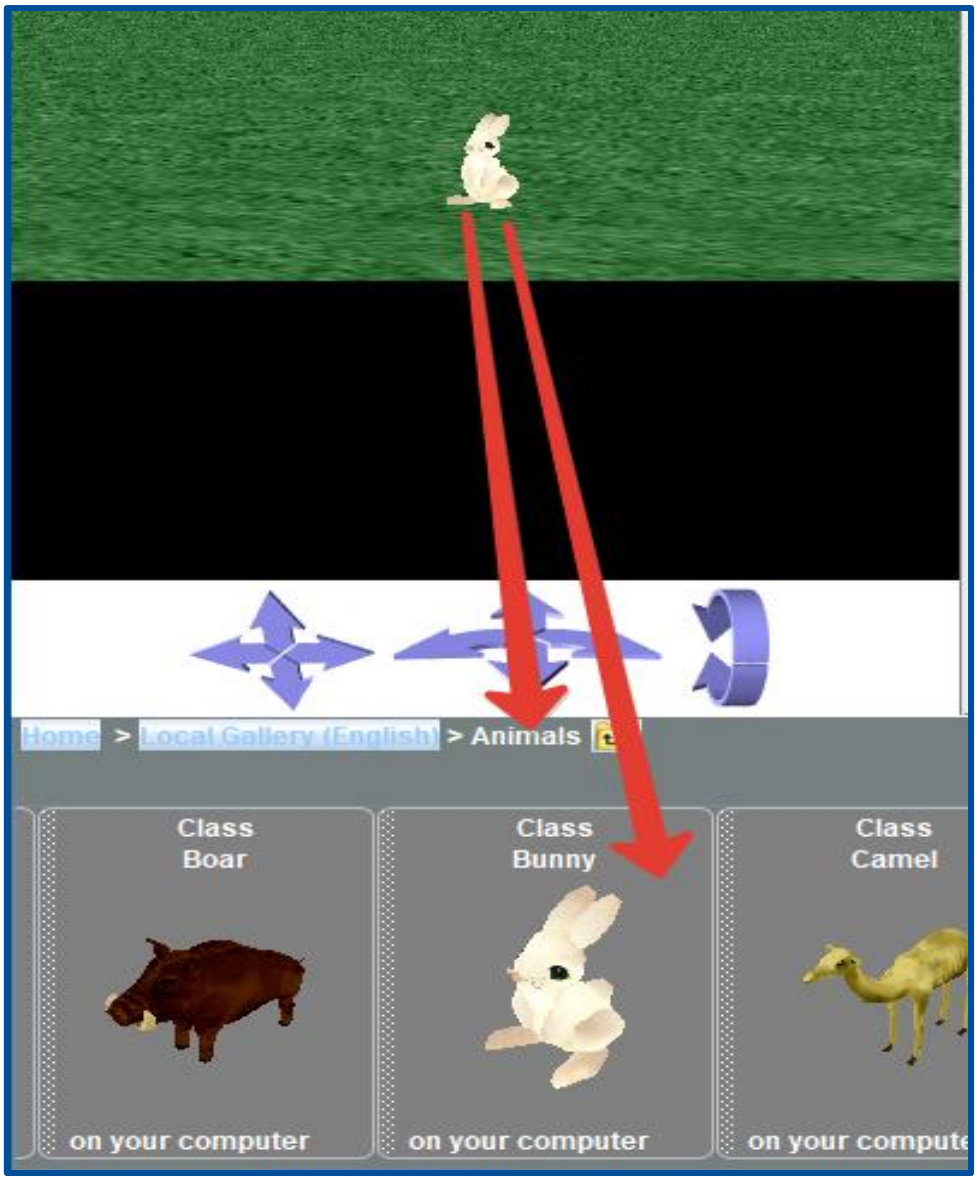


Рис. 119. Добавление зайчика

3. Объект можно перенести в мир простым перетаскиванием мышью. Можно нажать правую кнопку мыши при наведении курсора на объект, а затем в появившемся диалоговом окне нужно нажать кнопку добавления объекта в мир – **Add Instance to World**. Зайчик появится в вашем мире.

Нажмите на зеленую кнопку **Done** (готово), чтобы начать работу с объектом-зайчиком.

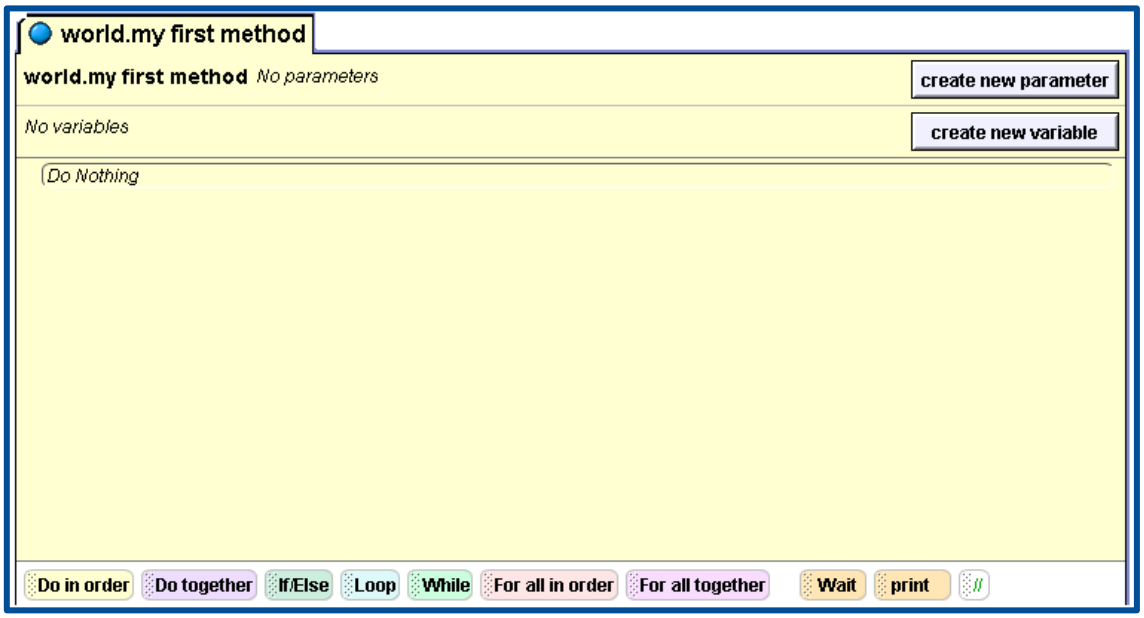


Рис. 120. Окно Method Editor

4. На экране Alice вы должны увидеть **Method editor** (редактор методов). Это область, где вы будете говорить вашему зайчику, что делать (рис. 120).

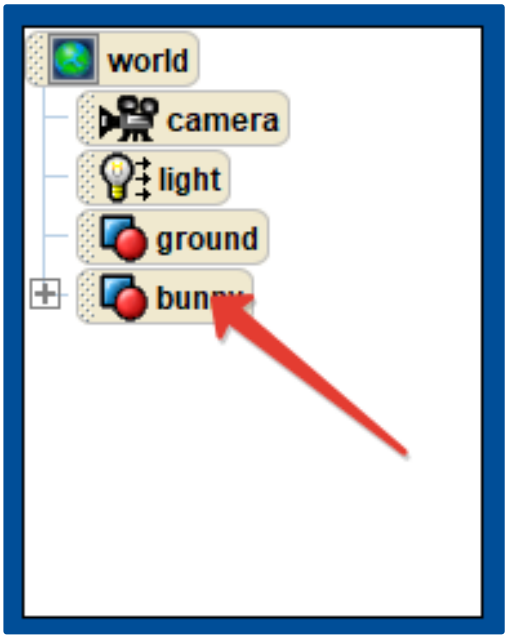


Рис. 121. Окно Object Tree

Метод (method) – команда, которую вы можете сообщить своему зайчику. Ваш зайчик уже знает определенные команды. Найдите **Object Tree** (дерево объектов), которое включает объекты в вашем мире, в левом углу экрана и нажмите на зайчике (Bunny) (рис. 121).

5. Добавим в код какие-нибудь действия для нашего зайчика. Зайчик уже знает некоторые команды (методы). На рис. 122 показаны некоторые методы зайчика.

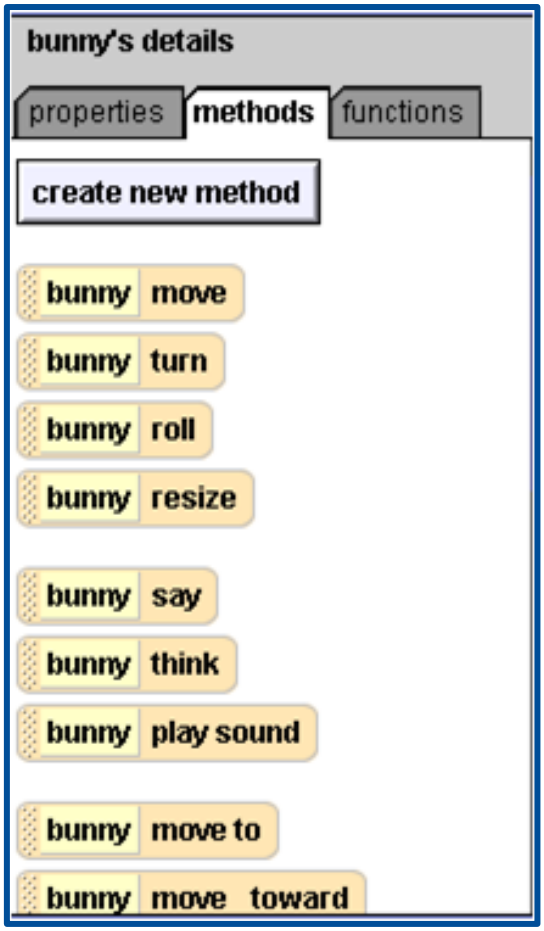


Рис. 122. Окно методов

Найдите кнопку **Do in order** (делай по порядку) в нижней части редактора метода и перетащите в редактор метода. Теперь, когда мы перетаскиваем методы внутрь этого метода **Do in order**, они будут выполняться один за другим в том порядке, в котором они перечислены (рис. 123).

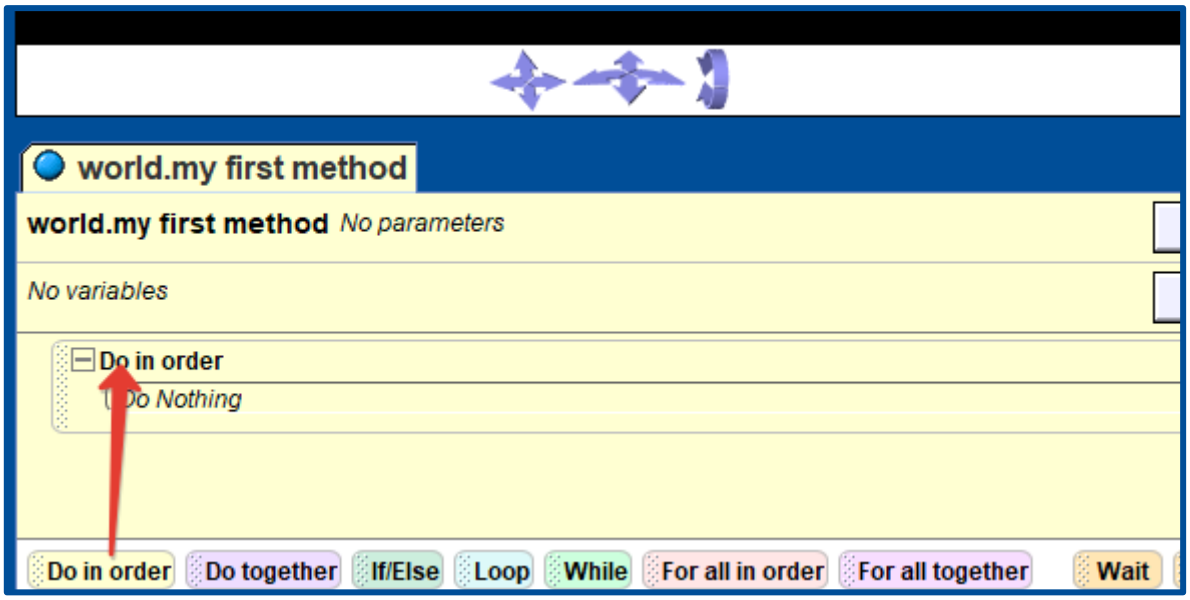


Рис. 123. Создание алгоритмической конструкции Do in order

В дереве объектов кликните по объекту **Bunny**. В окне методов нажмите на метод **bunny.move** (метод передвижения) и перетащите его в окно кода в **Do in order** (рис. 124).

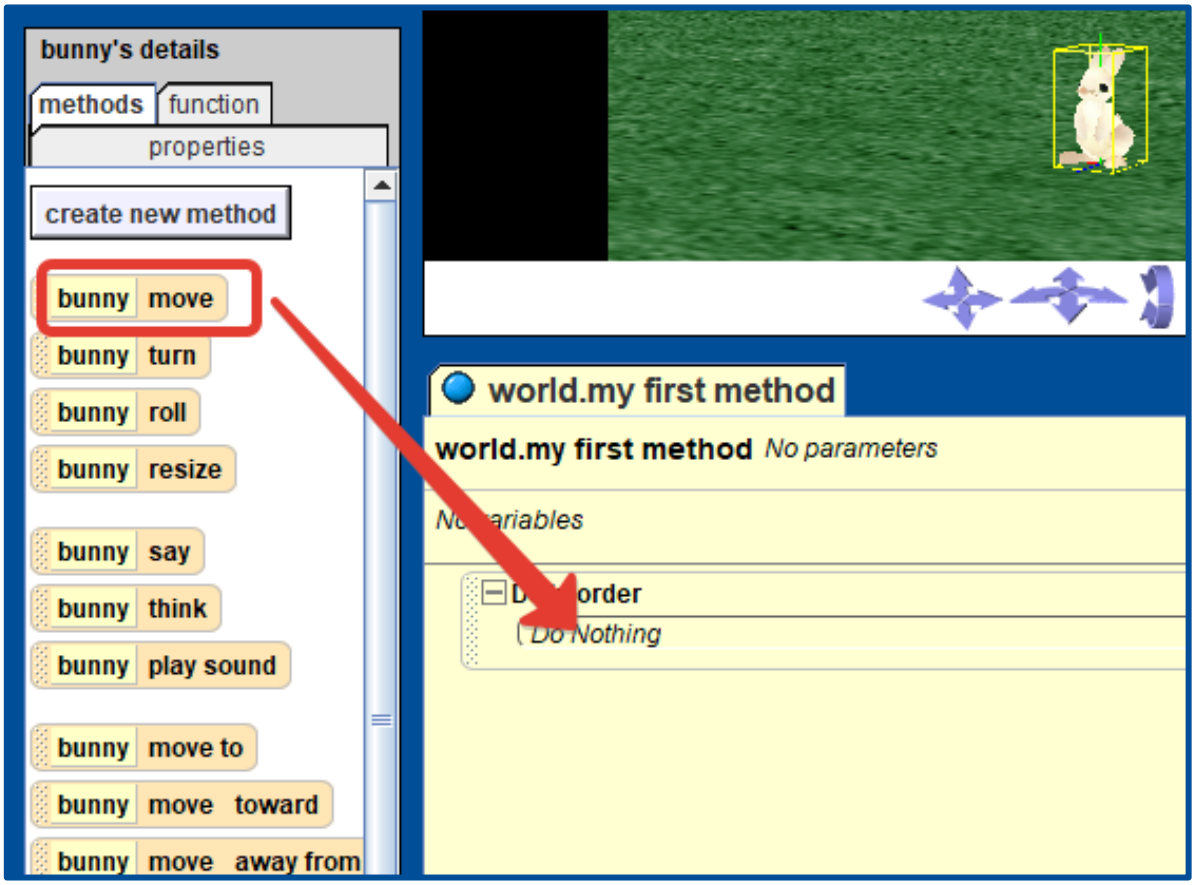


Рис. 124. Перемещение метода движения для зайчика

6. Метод move имеет свои особенности: объект может двигаться в определенную сторону (**up** – вверх, **down** – вниз, **left** – налево, **right** – направо, **forward** – вперед, **backward** – назад). Когда вы отпустите метод **move**, вы увидите несколько вариантов для вашего метода, которые включают направление (direction) движения объекта и расстояние перемещения. Нам нужно, чтобы зайчик двигался вверх, для этого наведите курсор мыши на направление (direction) – **up**, а затем из выплывшего меню выберите 1 meter (рис. 125).

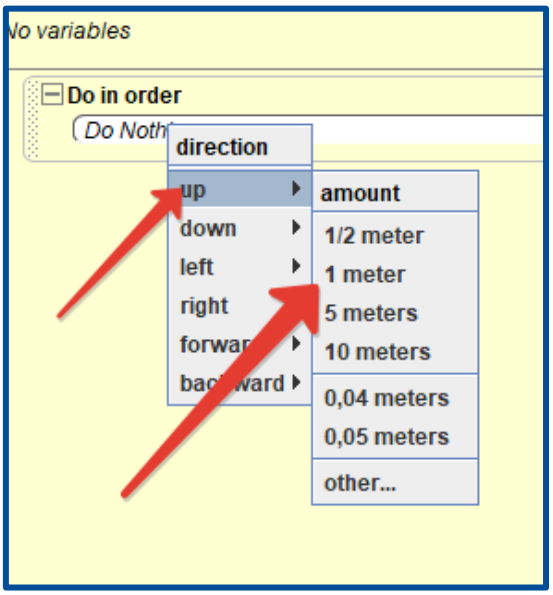


Рис. 125. Добавление параметров для метода move

Ваша первая команда для зайчика завершена. Он прыгнул вверх.

Теперь нужно, чтобы зайчик прыгнул вниз. Для этого также используем метод **move**, но только двигаться нужно вниз (**down**). Можно ускорить составление алгоритма: щелкните правой кнопкой мыши по вашей команде **move** и выберете копирование (**make copy**). Ваша команда передвижения (**move**) будет скопирована, под той, которую перетащили. Нажмите на маленькую стрелку внизу у второй команды и измените вверх (**up**) на вниз (**down**). Ваш редактор метода должен выглядеть, как на рис. 126.

Найдите кнопку 

в левом верхнем углу экрана и нажмите на нее, чтобы посмотреть, как ваш зайчик прыгает.

Рис. 126. Добавление движения вниз

7. Теперь вы узнаете, как избавиться от методов, которые вам не нужны. Допустим вы не хотите, чтобы зайчик двигался вниз. Нажмите в мире на **move** в вашей команде **bunny move down** и перетащите ее в корзину в левой верхней части окна. Когда контур корзины будет зеленым, вы можете отпустить команду, чтобы удалить ее (рис. 127).

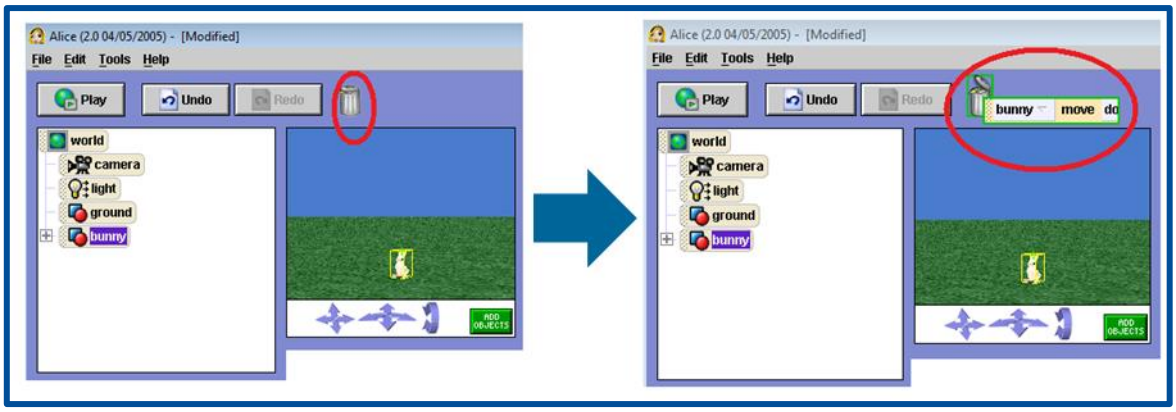


Рис. 127. Удаление ненужных методов

Если вы удалили что-то по ошибке или что-то сделали неправильно можно отменить команду, нажав кнопку **Undo** (Отменить) в верхнем левом углу экрана (рис. 128).

Попробуйте нажать кнопку **Undo**, чтобы вернуть команду **bunny move**, которую вы только что удалили.

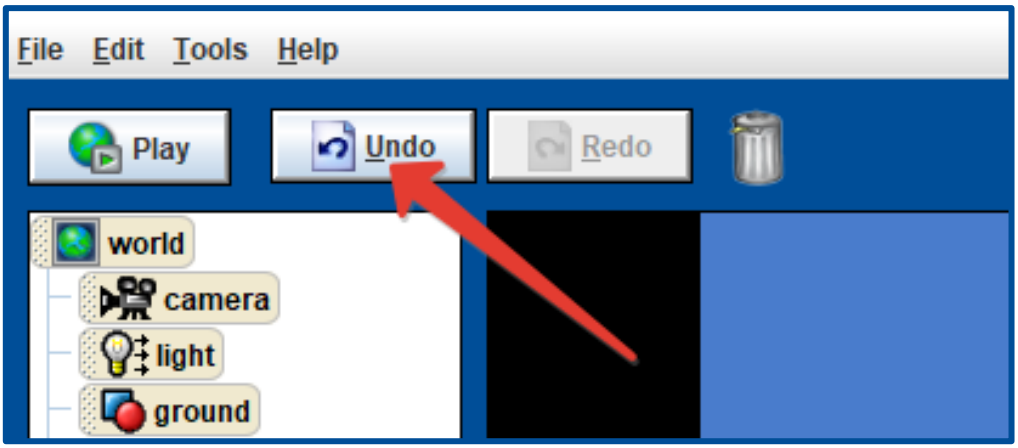


Рис. 128. Отмена удаления

8. Сделаем так, чтобы зайчик смог выполнить два действия одновременно: шевелить ушами и что-то говорить. Для этого воспользуемся алгоритмической конструкцией **Do together**. Кнопка для конструкции Do together расположения внизу окна кода. Найдите и перетащите эту конструкцию в свой код (рис. 129).

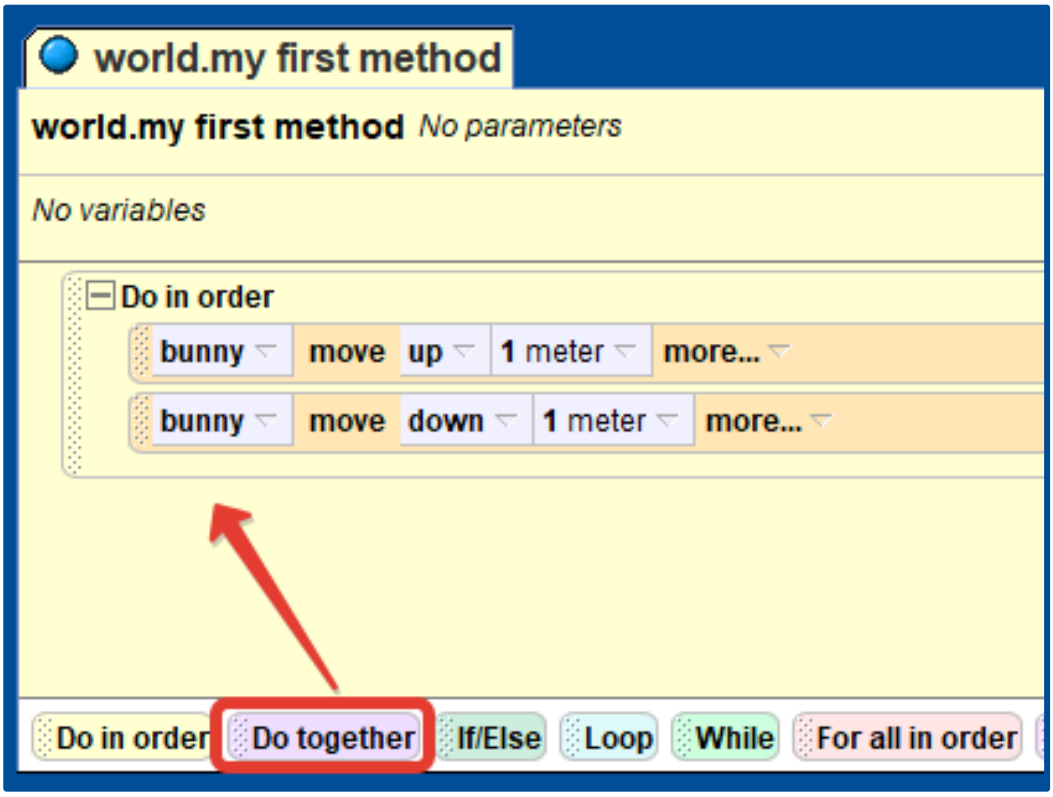


Рис. 129. Создание алгоритмической конструкции Do together

9. Создадим методы для движения ушек зайчика. Зайчик – сложный объект. В дереве объектов у объекта Bunny рядом есть плюс , нажав на который вы можете увидеть сколько у зайчика составных частей. У него есть **rightLeg** (правая нога), **leftLeg** (левая нога), **upperBody** (верхняя часть туловища), **tail** (хвост). У **upperBody** в свою очередь есть такой же плюс . Ушки относятся к **head** (голова) и имеют названия **rightEar** (правое ухо), **leftEar** (левое ухо). Раскрытый список объекта bunny показан на рис. 130.

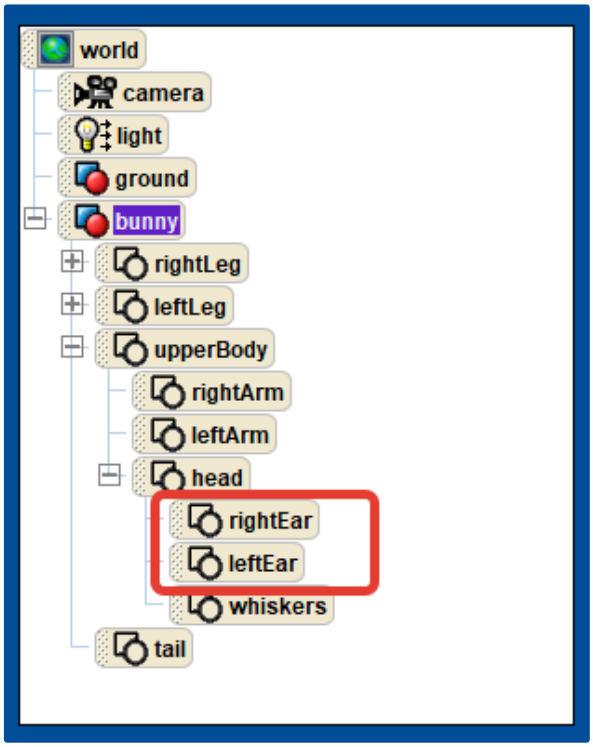


Рис. 130. Подобъекты объекта bunny

Чтобы зайчик шевелил ушами, будем использовать метод **turn** (поворот). Для того чтобы видеть методы ушей зайчика, щелкните **rightEar**, и они будут отображаться в левом нижнем углу экрана. Выберите метод поворота (**turn**) и тащите до вашей команды Do Together в редакторе метода. Чтобы ухо поворачивалось, выберете **left** в направлении (direction) и затем выберете **1/4 Revolution** (0,25 оборота, 90 градусов), чтобы ухо поворачивалось влево (рис. 131).

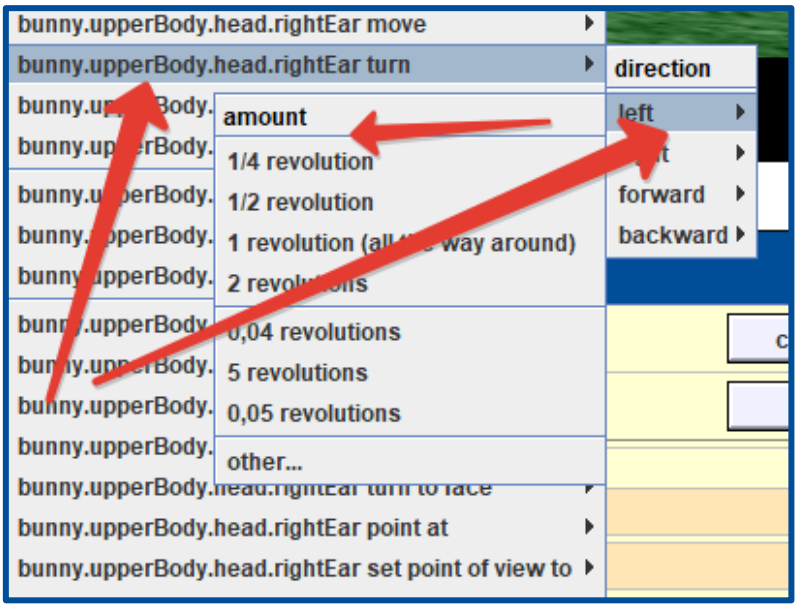


Рис. 131. Метод turn для уха зайчика

Теперь повторите процесс для **leftEar**, только поворот уха сделайте направо (**right**) (рис. 132).

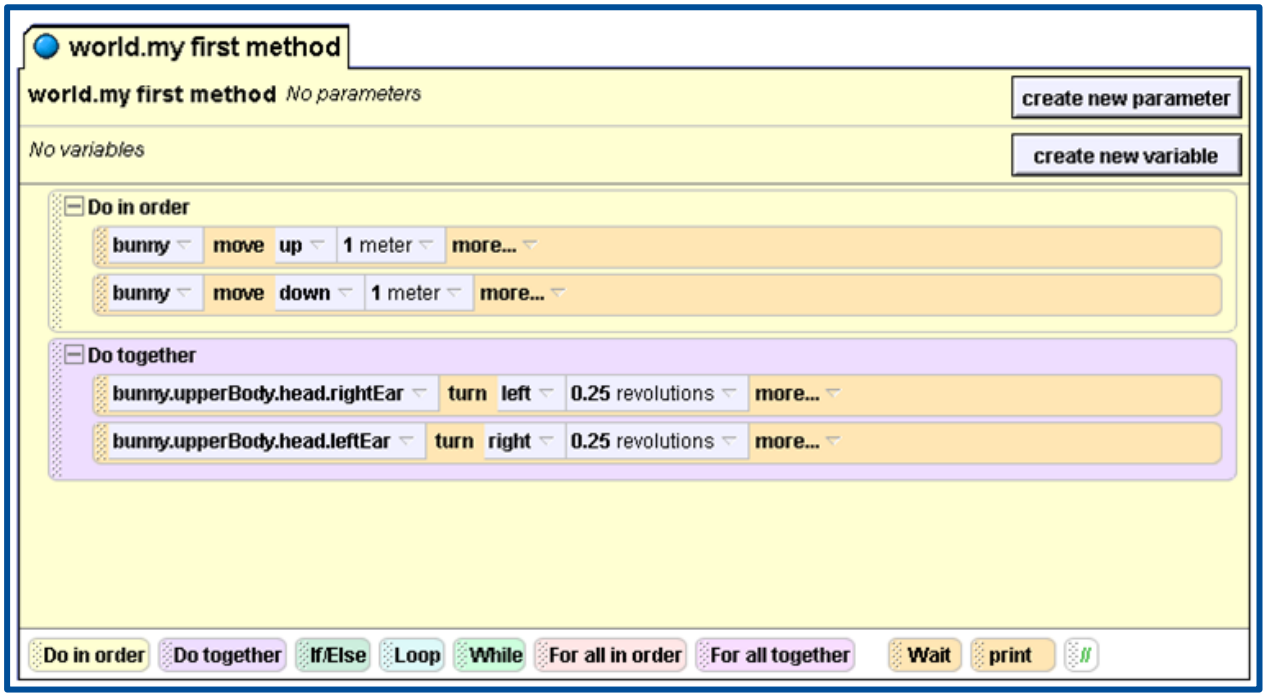


Рис. 132. Окончательный алгоритм шевеления ушами зайчика

10. Теперь, когда ваш зайчик шевелит ушами, нужно сделать, чтобы он говорил. Нажмите на изображение зайчика (Bunny) в дереве объектов для

отображения методов зайчика. Затем нажмите bunny Say (говорить) и перетащите в вашу команду Do Together под другими методами. Появится небольшое меню:

Рис. 133. Диалоговое окно метода bunny say

Нажмите other и введите текст «Hello, world!» («Привет, мир!»). Нажмите кнопку 

, чтобы посмотреть, как выглядит ваш мир сейчас.

11. Вы должны заметить, что когда вы запустили ваш мир, то фраза зайчика появилась и исчезла очень быстро, вы едва ли успели ее прочитать. Но есть способ исправить это. Посмотрите в строке редактора метода на команду, по которой ваш зайчик говорит. На этой линии щелкните список **more…** рядом с командой (рис. 134).

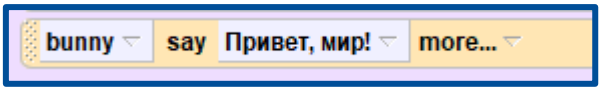


Рис. 134. Опция More в конце команды

Нажмите на продолжительность (**duration**) в небольшом меню, которое появится. Вы можете видеть, что выбрана 1 секунда (1 second) – это продолжительность появления облака с речью. Если мы хотим, чтобы облако с речью находилось на экране дольше, щелкаем **other**, и на калькуляторе, который появляется, вводим **3** (рис. 135).

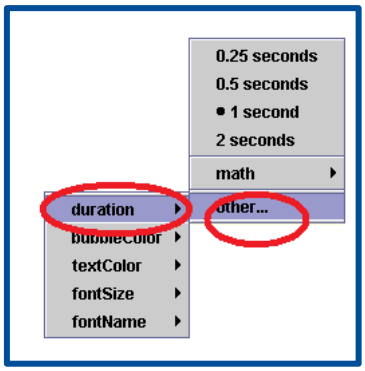


Рис. 135. Установка продолжительности действия команды

Теперь запустите ваш мир. Вы успеваете прочитать речь зайчика! Поздравляем с завершением создания первого мира в Alice!

Самостоятельное задание. Создайте другие команды для движения зайчика, например, повороты вокруг оси (**turn** или **roll**), думать (**think**), поворот лицом к (**turn to face**), изменение размеров (**resize**) и др. Вы только посмотрите, что вы можете заставить его сделать!

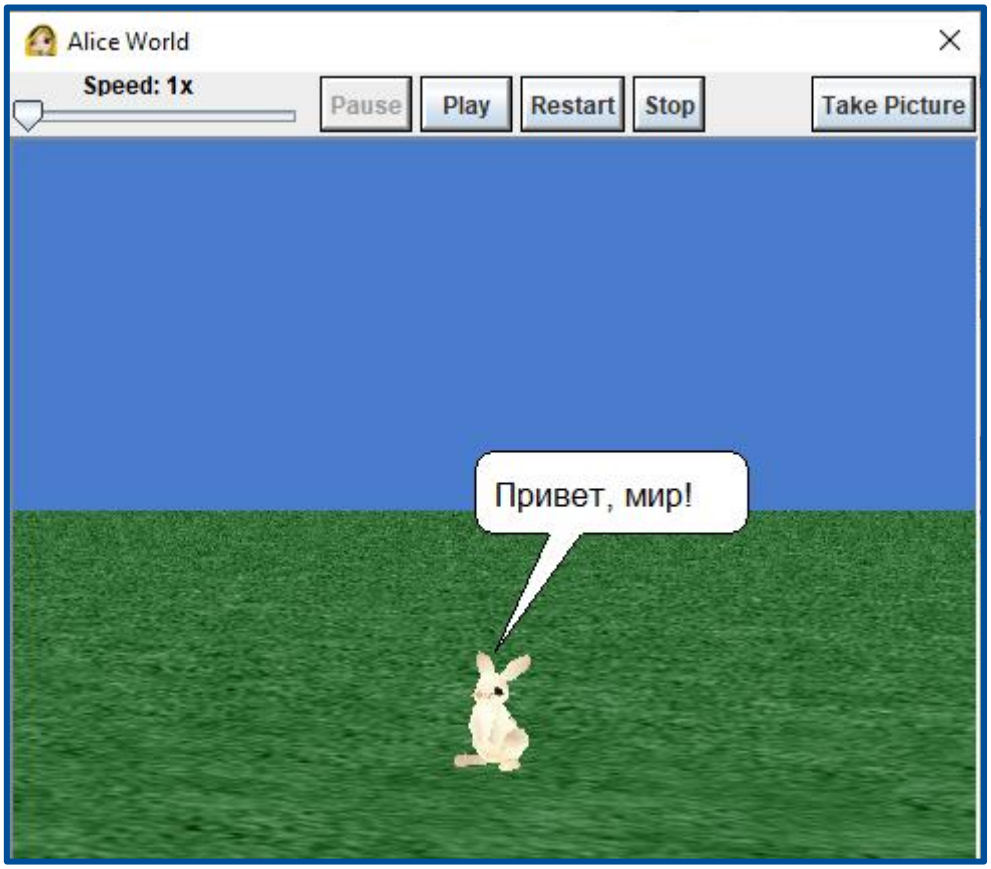


Рис. 136. Итоговый проект

2.2. Лабораторная работа 2. Управление объектами в Alice

Задание 1. Разработка проекта соревнования кенгуру и черепахи

В Alice можно организовать взаимодействие нескольких объектов. Как мы уже узнали ранее, у объектов есть заранее определенные методы (move, turn, say и др.). Но также можно создавать и собственные методы для объектов. В этой лабораторной работе мы познакомимся с оператором цикла Loop, с управлением несколькими объектами, созданием собственных методов [3].

Разработаем проект, в котором черепаха и кенгуру соревнуются друг с другом. Нам нужно очень хорошо подумать, как должна двигаться черепаха и как может прыгать кенгуру. А как вы думаете, кто будет первым?

1. Создайте новый мир (File – New World). Выберите понравившуюся локацию мира. С помощью кнопки **Add Objects** добавьте на сцену наших персонажей: кенгуру (**Kangaroo**) и черепаху (**Turtle**).

2. У мира уже есть стандартный метод **my first method**, который запускается, когда стартует мир (запускается программа кнопкой **Play**). Создадим еще один метод **race**. Для этого в дереве объектов кликните **world** (мир). В окне методов нажмите на кнопку **create new method** и в появившемся диалоговом окне введите **race** (рис. 137).

3. В окне методов рядом с новым методом **race** находится кнопка **edit.** Нажмите на нее, откроется окно метода **race** для написания кода.

Разработайте код поведения черепахи и кенгуру по следующему алгоритму:

Делай по порядку
- черепаха поворачивается направо на 0,25 оборота;
- черепаха говорит: «Я обгоню тебя»;
- кенгуру говорит: «О'кей, давай попробуем»;
- черепаха поворачивается налево на 0,25 оборота.

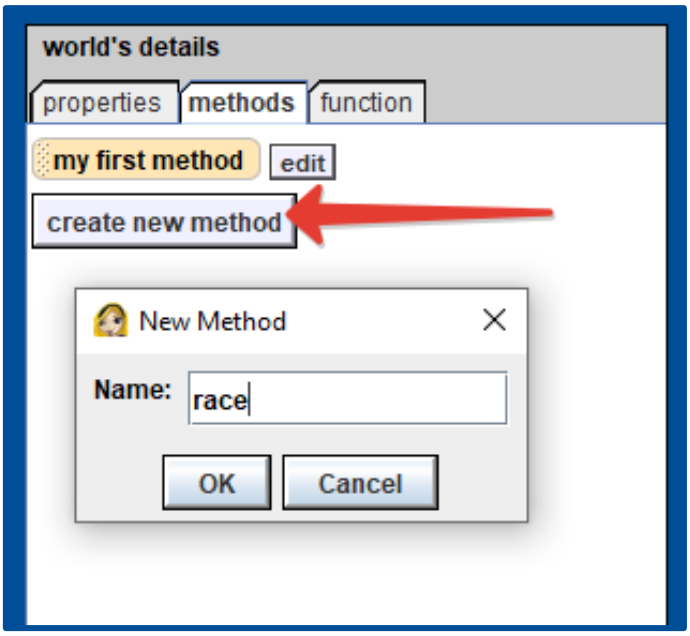


Рис. 137. Создание метода race

Запустите проект. Посмотрите, все ли работает так, как надо. Возможно, вы не увидите никакого действия. Мы не вызвали метод **race** в первом методе **my first method**. Перенесите метод **race** из окна методов в окно кода (рис. 138).

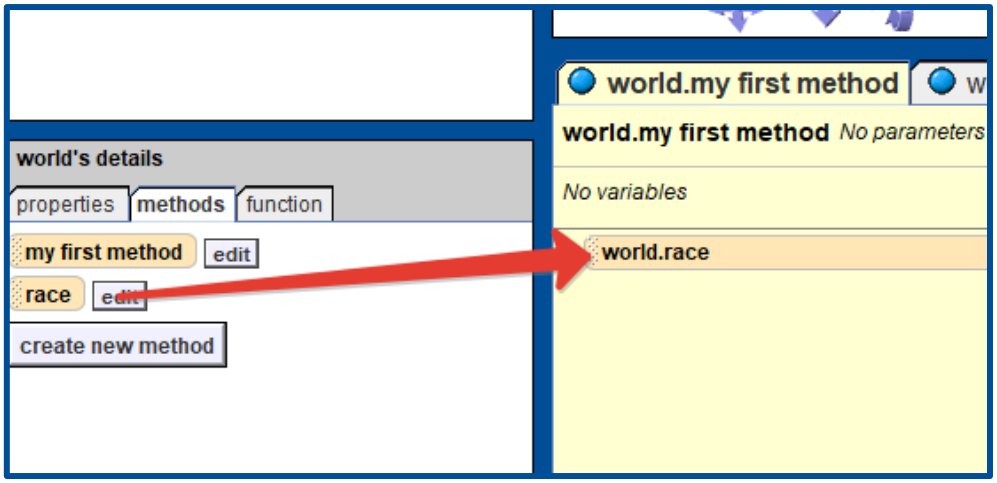


Рис. 138. Вызов метода race

Снова запустите проект. Проверьте, все ли теперь работает как надо. Вы использовали методы **turn** и **say**, не так ли?

4. Добавим в наш проект цикл из 4 шагов, в котором черепаха и кенгуру одновременно выполняют перемещение. Конечно, черепаха и кенгуру

двигаются по-разному: черепаха ползёт, а кенгуру совершает прыжки. Нам нужно придумать способы передвижения наших персонажей, всего один шаг. В цикле мы повторим их 4 раза. Таким образом у нас получится четыре последовательности передвижения.

Перенесите конструкцию Loop в ваш проект, после конструкции **Do in order** метода **race**. При переносе цикла у вас появится диалоговое окно, в котором нужно выбрать количество повторений. В нем выберите пункт **other** и с помощью калькулятора введите число **4**.

Создайте для черепахи метод **walk.** Например, черепаха движется вперед (**move forward**) на 0,2 метра. При этом она ползёт медленно: один шаг выполняет за 2 секунды (**duration 2 seconds**).

Создаем метод для кенгуру. Кенгуру делает высокий и быстрый прыжок, движется одновременно (**do together**) вверх (**up**) и вперед (**forward**), а потом также одновременно движется вперед и вниз (**down**). Весь прыжок (4 действия) занимает 1 секунду.

В результате, вызов методов **turtle.walk** и **kangaroo.hop** будут выглядеть, как показано на рис. 139.

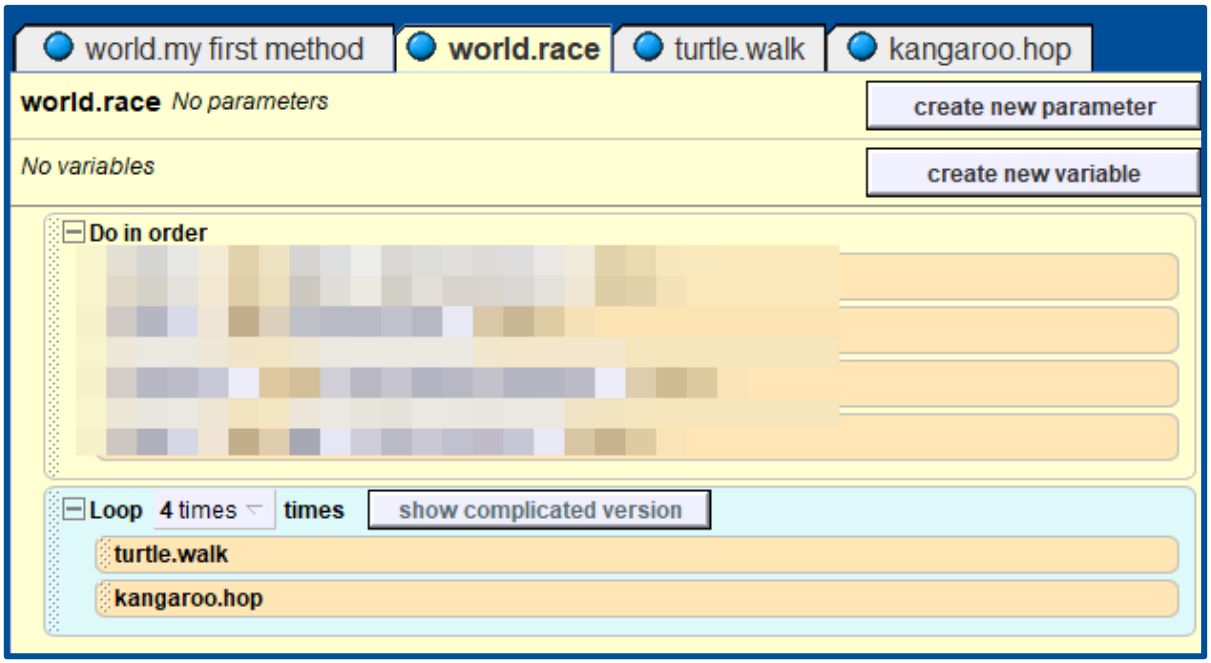


Рис. 139. Цикл из 4 шагов движения персонажей

5. Для фона добавьте объект **waterFall** (водопад), который находится в папке **Nature** (природа). Добавьте и другие объекты в окружении.

6. Придумайте историю окончания этой супергонки. Кто все-таки победил? Что сказали герои в конце гонки? Делали ли персонажи смешные движения? Может, кто-то был так рад победе, что кружился или прыгал на месте?

Дополните эту историю и покажите преподавателю.

Задание 2. Управление объектами с помощью клавиатуры и мыши

Созданным миром можно управлять с помощью клавиатуры и мыши. Например, с помощью клавиш «вверх» (↑), «вниз» (↓), «влево» (←), «вправо» (→) можно организовать управление камерой, клавиши на клавиатуре позволят вызвать методы управления объектами и др.

Создадим сцену с лягушкой, которая будет делать какие-то действия в зависимости от того, какую кнопку на клавиатуре мы нажмем.

1. Создайте новый мир. Придумайте локацию: прудик, озеро или водопад. Поместите на сцену лягушку (**frog**) из папки **animals**.

2. Вначале, когда стартует мир, сделаем так, чтобы лягушка «лицом» повернулась к камере. Для этого для лягушки вызовите метод **turn to face** и выберите объект **camera** (рис. 140).

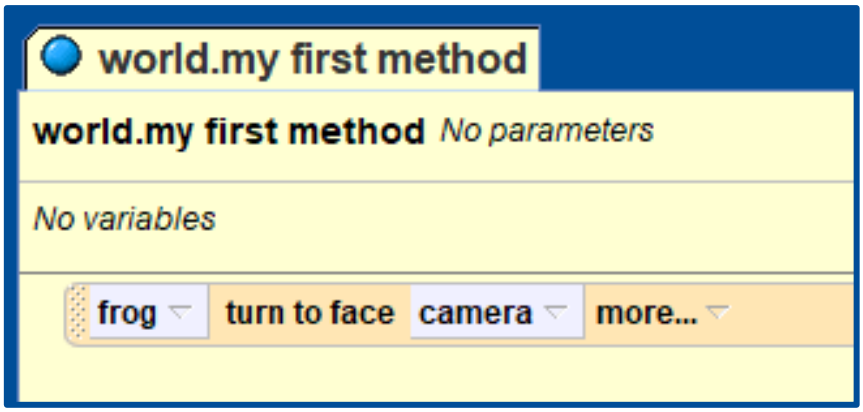


Рис. 140. Поворот лягушки к камере

3. У нашей лягушки уже есть несколько готовых методов: **foottap** (шлепок лапой), **ribbit** (квакание), **headnod** (кивок головы). Вызовем эти методы по нажатию соответственно клавишам 1, 2 и 3.

Для этого в правом окне нажмите на кнопку **create new event**. В появившемся выпадающем списке выберите **When a key is typed** (когда нажата клавиша). Появится заготовка метода. В поле **any key** выберите **Num1** (клавиша 1 на цифровой клавиатуре), в поле **Nothing** выберите метод для лягушки **foottap** (рис. 141).

Рис. 141. Создание события по управлению клавишами

Запустите проект. Нажмите на 1 на цифровой части клавиатуры. Что произошло?

4. Сделайте вызовы методов кваканья и кивка головы лягушки по нажатию на другие клавиши.

5. Добавим возможность управления камерой с помощью клавиш управления курсором (вверх ↑, вниз ↓, влево ←, вправо →). Создайте событие **Let the arrow keys move subject** (позволить стрелками управлять предметом). В качестве объекта выберите **camera**. Запустите проект, попробуйте понажимать стрелки. Вы управляете созданным миром.

6. Создайте для лягушки методы прыжков влево и вправо, вверх и вниз. Вызовите эти методы с помощью клавиш WASD (W – вверх, A – влево, S – вниз, D – вправо).

7. Для лягушки создайте метод **Сияние** (методы можно называть и русскими буквами). С помощью кнопки **create new method** создайте новый метод. Перейдите на вкладку **properties** (свойства). Раскройте список **Seldom Used Properties** (редко используемые свойства) и перенесите свойство **emissiveColor** (цвет сияния) в окно кода (рис. 142). Измените цвет на желтый.

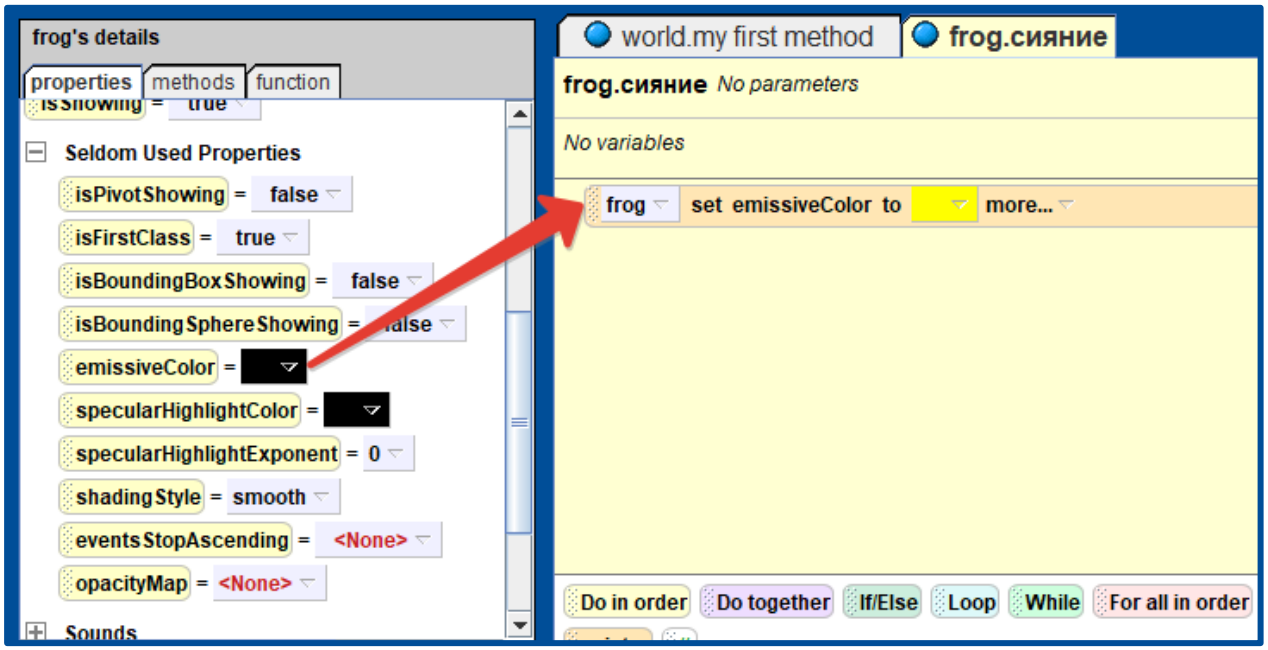


Рис. 142. Создание метода Сияние

Сделайте вызов данного метода при клике мышью по лягушке (**When the mouse is clicked on something**).

Запустите мир, проверьте его работу и покажите преподавателю.

2.3. Лабораторная работа 3. Алгоритмы ветвления в Alice

В этой лабораторной работе мы познакомимся с условной конструкцией. Алгоритмическая конструкция If/else расположена внизу окна кода. Она используется в тех случаях, когда нужно выполнить какие-то действия в зависимости от некоторого условия. Например, если что-то **true** (истина), то объекты

совершают одну последовательность действий: если какое-то уловие **false** ложь, то объекты совершает другую последовательность действий.

Задание 1. Бабочка просит пользователя ввести число. Если пользователь ввёл число от 1 до 3, то размеры бабочки увеличиваются на это число. Если же пользователь ввёл что-то вне этого диапазона, то бабочка уменьшается в размерах

1. Создайте новый проект и поместите в него **Butterfly** (бабочку) из папки Animals/Bugs (рис. 143).

Рис. 143. Класс Бабочка

2. Возможно, ваша бабочка имеет маленькие размеры. Давайте ее увеличим. В режиме нажатой кнопки **Add objects** увеличьте размеры бабочки с помощью кнопок (рис. 144).

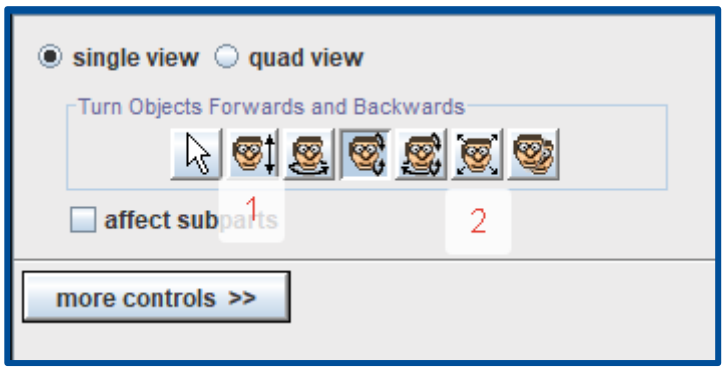


Рис. 144. Кнопки изменения размера и положения объекта

3. Добавьте к вашему миру деревья и другие объекты. Когда закончите с объектами, посмотрите, что получилось (рис. 145).

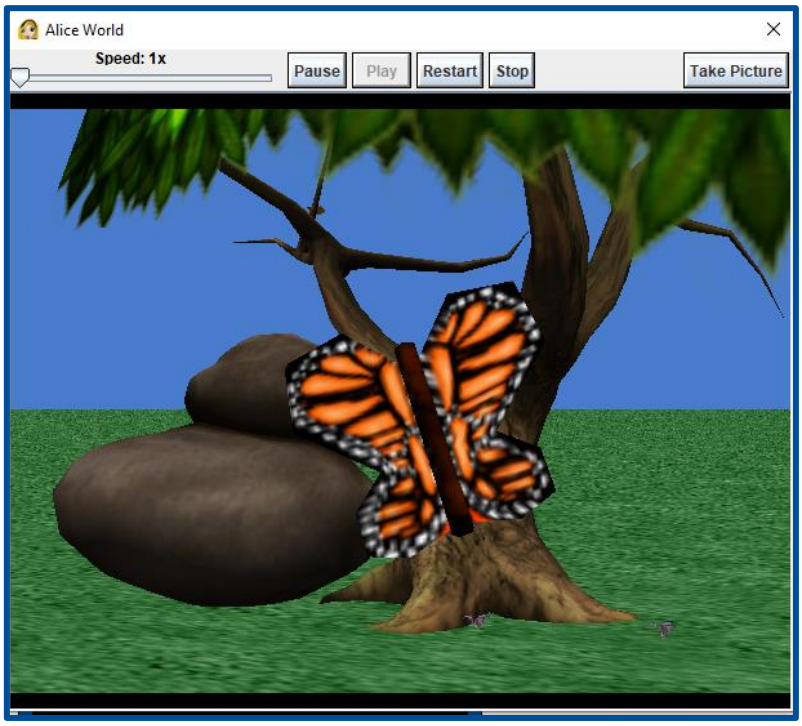


Рис. 145. Вид проекта с объектами

4. Для нашего проекта нам потребуется где-то хранить данные пользователя. Например, пользователь ввёл число 2, и в зависимости от этого значения у нас будет происходить проверка условия принадлежности диапазону от 1 до 3. Для того чтобы сохранять некоторые данные, нужны переменные. Переменные бывают различных типов: числа, строки, списки и так далее. Для нашей задачи нам потребуется переменная числового типа.

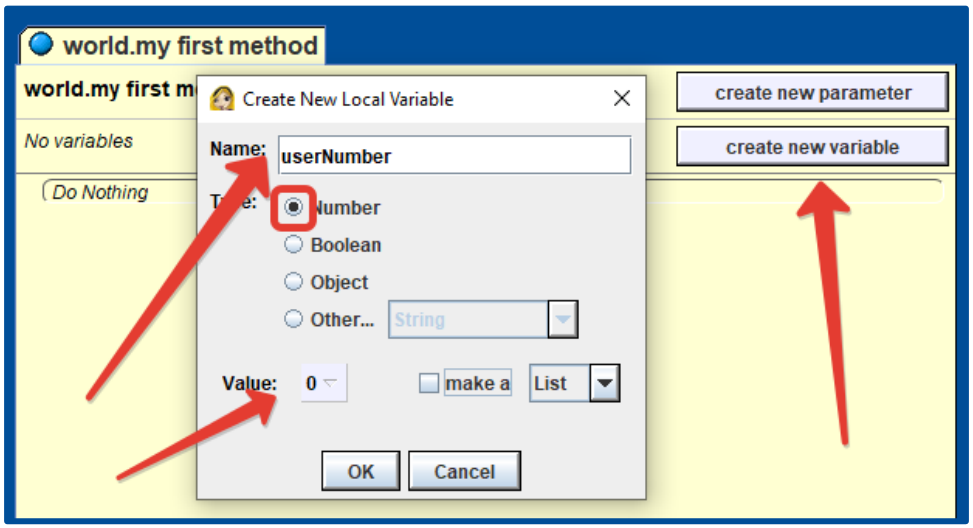


Рис. 146. Создание переменной userNumber

Создадим переменную **userNumber** (число пользователя). Для этого в окне кода нажмите кнопку **Create new variable** (создать новую переменную), выберите для неё тип **number** (число) и значение по умолчанию **0** (рис. 146).

5. Перетащите переменную **userNumber** в окно кода. У вас появится диалоговое окно с первым выбранным значением **set value** (установить значение). Выберите в выпадающем списке пока любое значение, мы его всё равно изменим (рис. 147).

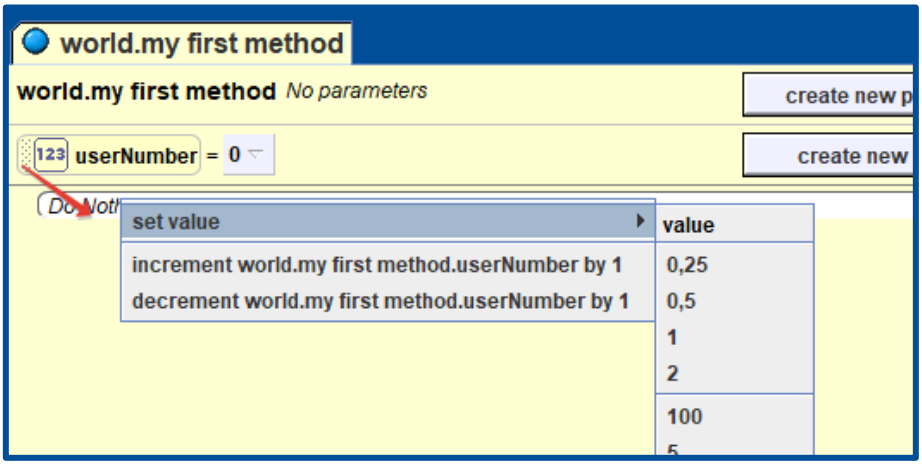


Рис. 147. Присваивание переменной значения

6. Вместо выбранного по умолчанию значения сделаем так, чтобы число было введено пользователем. Выберите в дереве объектов **world** и перейдите на вкладку **functions** (функции). Среди функций найдите функцию **ask user for a number** (рис. 148). Выбранную функцию поставьте вместо первоначального значения.

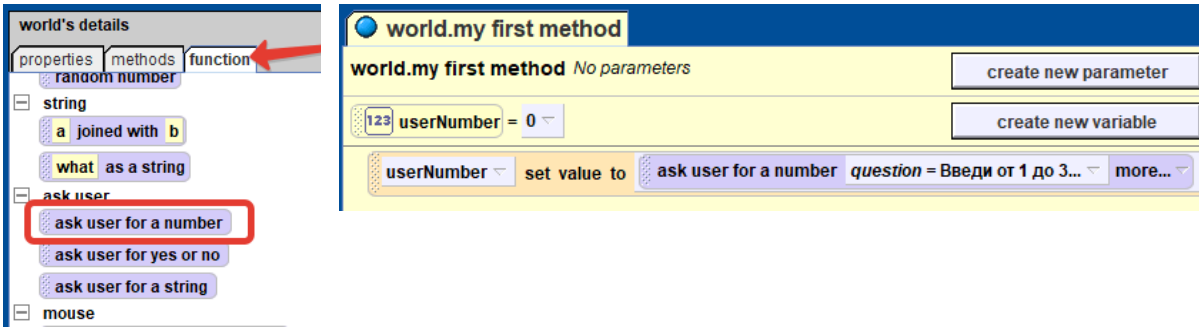


Рис. 148. Добавление функции ask user for a number

7. Перенесите алгоритмическую конструкцию **if/else** в окно кода. По умолчанию можно выбрать условия **true**. Проверьте условия нашей алгоритмической конструкции. Если **userNumber == 1**, то бабочка говорит: «Я вырасту в 2 раза». И вырастает (**resize**) в 2 раза. Перетащите нашу переменную **userNumber** вместо **true,** в выпадающем списке выберите **userNumber ==**, с правой стороны единица (рис. 149).

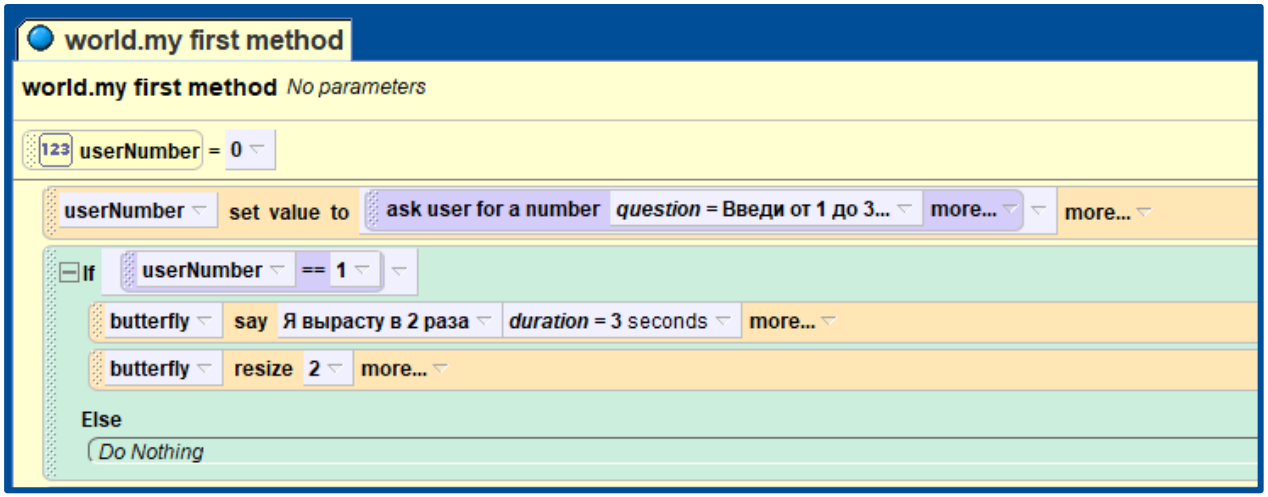


Рис. 149. Создание первого условия роста для бабочки

8. Аналогично добавьте условия для числа 2 (бабочка должна вырасти в 3 раза от первоначального размера) и для числа 3 (бабочка должна вырасти в 4 раза от первоначального размера). Если же пользователь ввел другое число, то бабочка уменьшается. Итоговый код программы показан на рисунке 150.

9. Закрепим наши знания по работе с условным оператором. Создадим проект, в котором змея называть число. В ответ на наш запрос змея будет сообщать, сколько колючек есть в пустыне. Например, если пользователь ввёл число 113, змея должна сказать: «113 колючек». Если пользователь ввёл число 3, то змея должна сообщать: «3 колючки». Таким образом, слова «колючек», «колючка», «колючки» выбираются в зависимости от того, какое число ввёл пользователь.

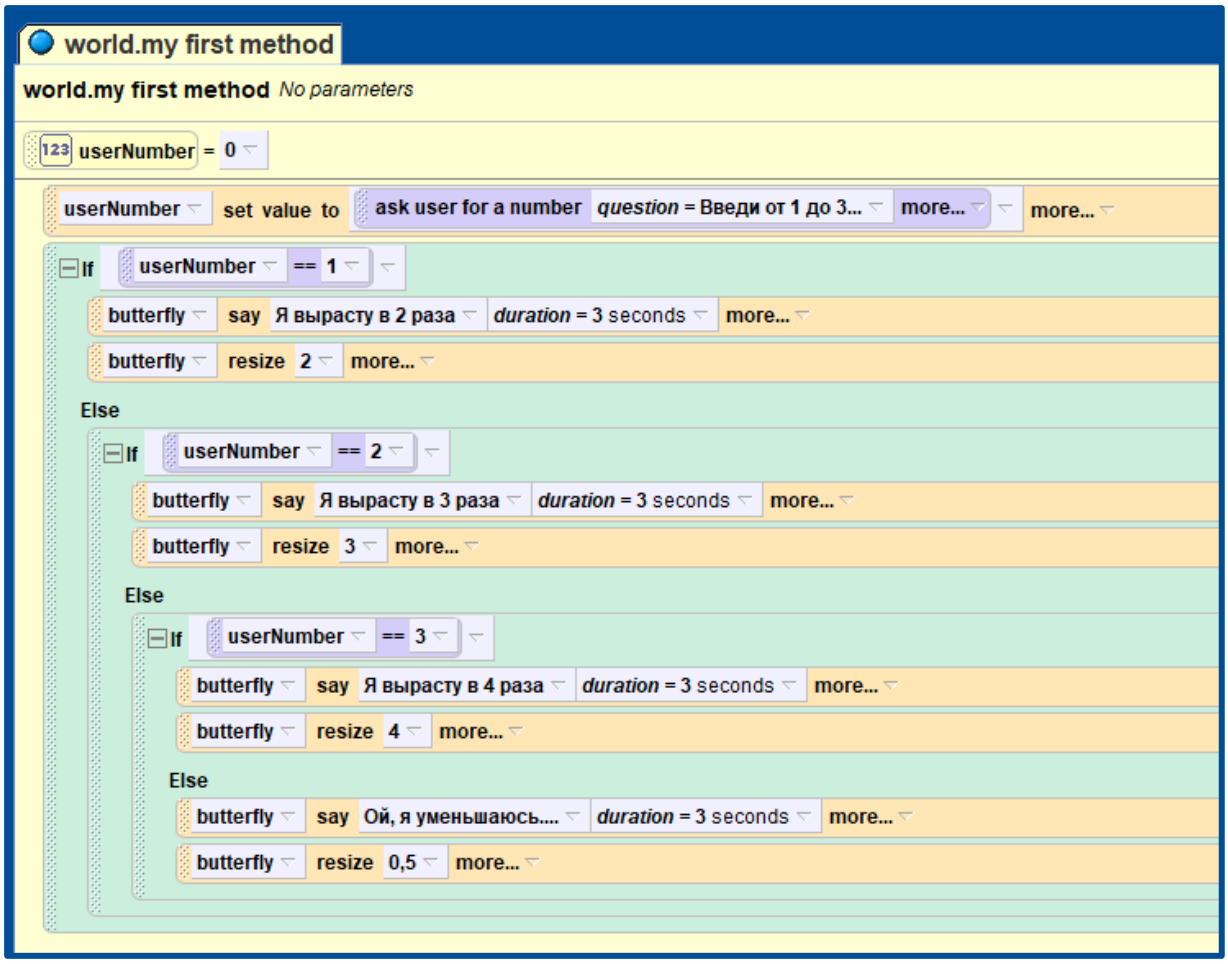


Рис. 150. Итоговый вид программы

Задание 2. Сколько колючек?

Алгоритм решения этой задачи заключается в следующем:

- Если число **number**, которое ввёл пользователь, при делении на **100** даёт в остатке число в диапазоне от **11** до **14** включительно, то змея сообщает: «**Number колючек**».
- Если при делении числа **number** на **10** в остатке получается **1**, то будет ответ «**Number колючка**».
- Если при делении числа **number** на **10** в числе получаются цифры **2**, **3** или **4**, то будет ответ «**Number колючки**».
- В остальных случаях будет **«Number колючек»**.

1. Создайте новый мир «песок» (**Sand**) и добавьте в него змею (**snake**) и объекты пустыни: кактусы и колючки из папки **Nature**.

2. В начале проекта вызовите имеющийся метод **slide** (ползти) для змеи. Затем с помощью метода **turn to face** разверните змею к камере.

3. Создайте две числовые переменные **number** и **ostatok**, начальное значение для каждой из них **0**.

4. Переменная **number** получает значение (**set value**) от пользователя с помощью функции **ask user for a number** с сообщением: «Укажи количество колючек».

5. Переменная **ostatok** получает целую часть (функция int a для объекта **world**) от деления **number** на **100**. Пусть **number = 123**, тогда эта операция должна дать **1**.

6. Переменная **ostatok** получает произведение **ostatok * 100**. Для первоначального **number = 123** здесь получим **100**.

7. Переменная **ostatok** получает целую часть (функция int a) от разности **number – ostatok**. Для первоначального **number = 123** здесь получим **23**.

8. Перенесите оператор **if/else.** Пусть первоначально будет условие **true**. Кликните по слову **true** в этой алгоритмической конструкции и в появившемся окне выберите **logic -> true or -> true** (рис. 151).

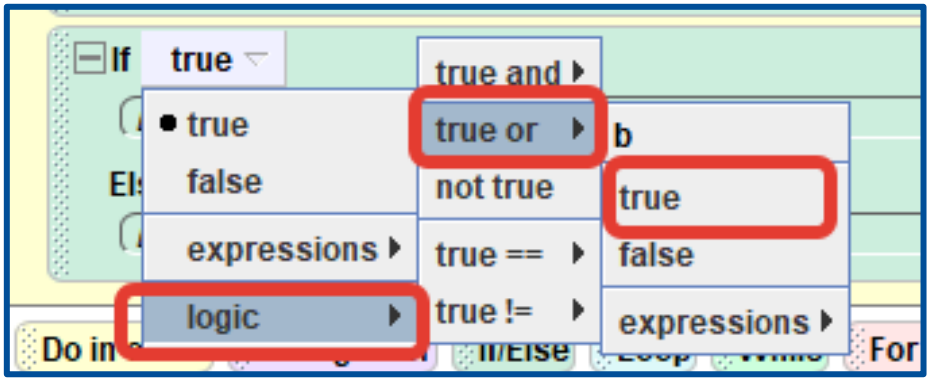


Рис. 151. Изменение условия

9. Кликните по первому **True** в условии и выберите условие **ostatok > = 11** (если числа 11 нет в списке, то укажите его через калькулятор в пункте **other**). Вид показан на рис. 152.

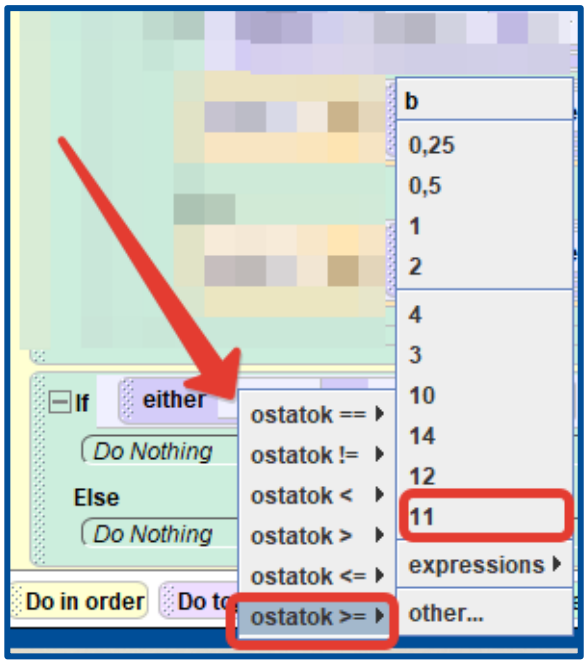


Рис. 152. Создание условия сравнения

10. Для второго **True** добавьте условие **ostatok <= 14**. Таким образом, у нас получилось условие, при котором выполняется или **ostatok = 11**, или **ostatok = 12**, **ostatok = 13**, **ostatok = 14**.

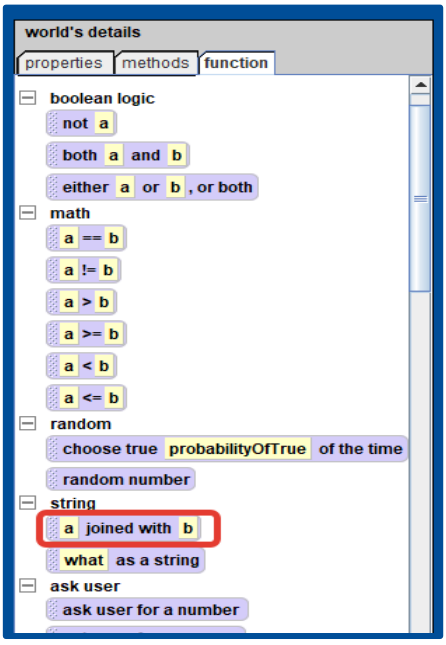


Рис. 153. Функция a joined with b

11. Для того чтобы змея сообщала, сколько колючек в пустыни, нужно объединить в одной строке и число, и текст. Сообщать обычный текст мы уже научились, а вот текст, который составлен из нескольких частей, требует

использования функции **a joined with b**. Ее можно найти на вкладке **function** для объекта **world** (рис. 153). Перетащите эту функцию в команду **snake say** вместо default string (строки по умолчанию).

12. Вместо первого параметра функции **a joined with b** перетащите функцию **what as a string** (что-то переводим в строку). Она нужна для перевода из числового формата в строку (рис. 154). Вместо **what** поставьте функцию **number** (в выпадающем списке она будет в разделе **expressions**). Дополнительно переменную **number** нужно еще перевести в числовой формат (рис. 155).

Рис. 154. Добавление функции what as a string

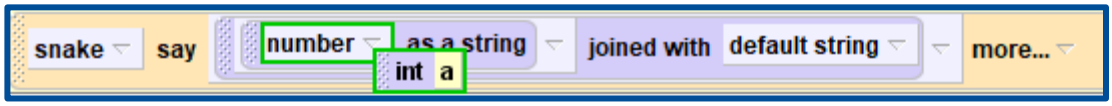


Рис. 155. Добавление функции int a

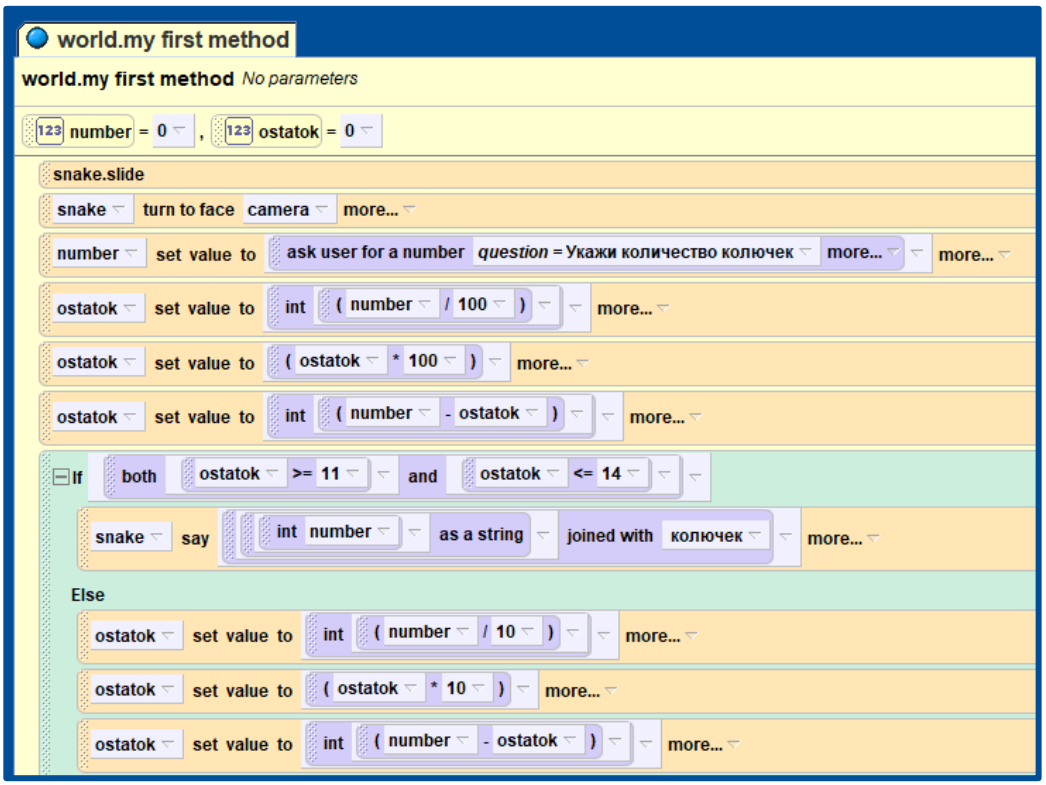


Рис. 156. Первая часть алгоритма «Сколько колючек»

13. Вторая часть этой строки будет содержать слово «колючек». На рис. 156 показано, как будет выглядеть эта часть алгоритма. Дополните его, чтобы змея правильно сообщала все варианты.

Задание 3. Сколько будет?

Самостоятельно разработайте проект, в котором какой-нибудь персонаж запрашивает у пользователя два числа: **number1** и **number2**. Затем персонаж спрашивает, какую операцию хочет выполнить пользователь: + (сложение), * (умножение), – (вычитание) или / (деление). В зависимости от того какой знак операции ввел пользователь, выполняется данная арифметическая операция.

Например, первое число (**number1**), которое вводит пользователь, **5**, второе число (**number2**), которое вводит пользователь **3**. Операция умножения (*).

В ответ персонаж должен сообщить: 5 * 3 = 15.

2.4. Лабораторная работа 4. Работа со списками и массивами объектов в Alice

В этой лабораторной работе поработаем со списками. Списки – это структуры данных, которые содержат несколько объектов (персонажей). Иногда необходимо сделать различным объектам повторяющиеся действия. Например, пяти объектам надо сделать поворот вокруг собственной оси. Вместо того чтобы для каждого отдельного персонажа программировать одно и то же действие, можно задать это действие для всех персонажей. Для этого персонажей объединяют в списки и с помощью циклов **for all in order** (для всех по порядку) и **for all together** (для всех вместе) можно сделать так, чтобы все объекты списка выполняли определённые действия в этом цикле или последовательно друг за другом или все вместе. Такой

подход даст серьёзную оптимизацию алгоритма и сделает процесс программирования ещё более легким и творческим.

Задание 1. Плавающие лодки и летающие аэропланы

1. Создайте новый проект, не забудьте его сразу же сохранить. Необходимо периодически сохранять проект.

2. С помощью кнопки **Add objects** в мир поместите **oasis** (оазис, вкладка Environments), разные деревья (вкладка Nature).

3. Добавьте **happySky** (небо, вкладка Environments/Skies) и настройте его расположение неба над оазисом. Если оно получилось немного серым, и не таким «happy», то добавьте на сцену источник освещения **lightBulb** (вкладка Lights). Ваша сцена сразу станет веселее.

Если какие-то из объектов найти не получается, то можно воспользоваться поиском по объектам, как показано на рис. 157.

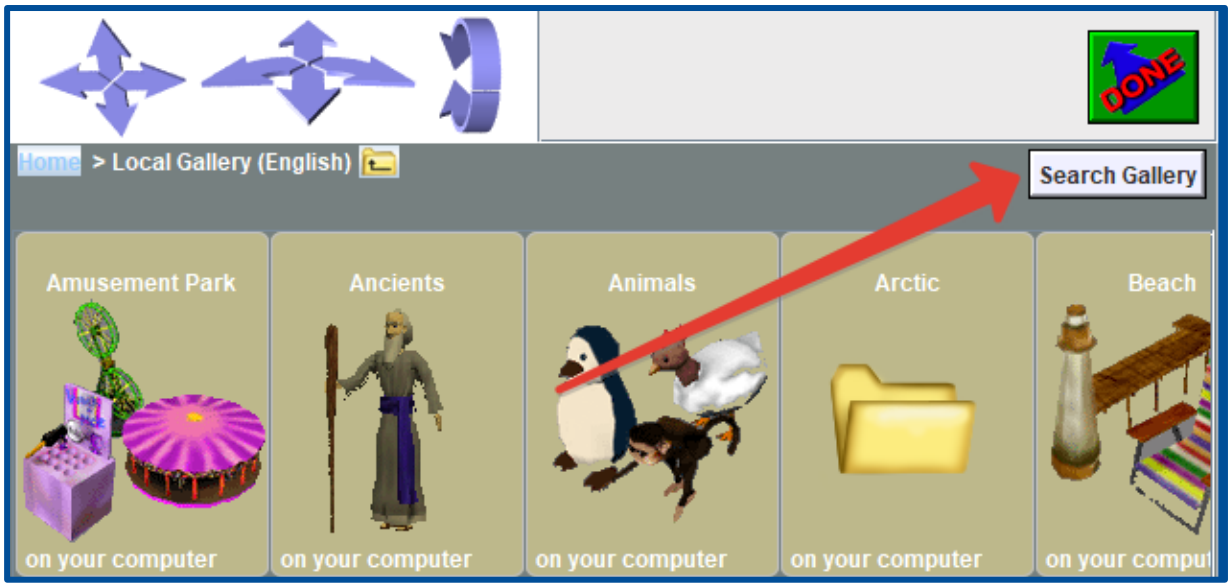


Рис. 157. Поиск по объектам в Alice

4. Разместим в нашем мире главных персонажей. Лодки (sailboat и raft) расположите на водной поверхности, а аэропланы (seaplane и t3A) над вершинами гор. Примерный вид мира показан на рис. 148.

Рис. 158. Примерный вид мира

5. Перейдем к коду. Нажмите кнопку **create new variable** и создайте новую переменную список **aList**, тип **Object** и установите галочку **make a List** (рис. 159)**.**

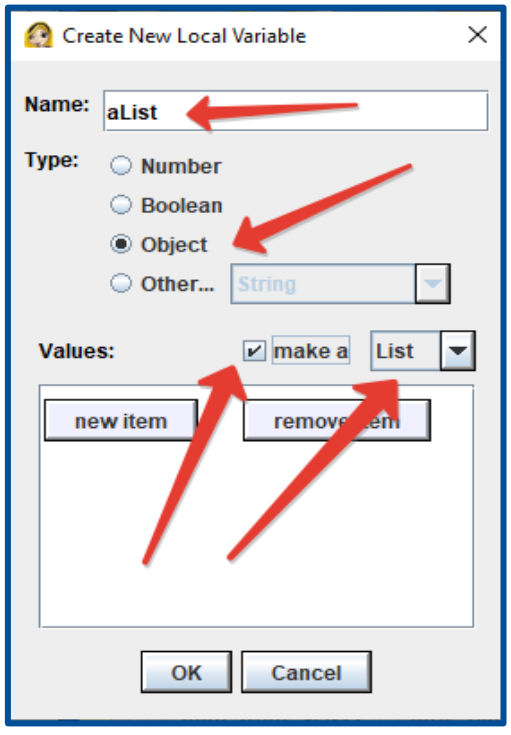


Рис. 159. Создание списка

6. Чтобы добавление объектов не приводило к задержке работы алгоритма, нажмите после каждого метода **insert** на опцию **more** в конце строки. Выберите duration = 0 (задержка выполнения команды 0 секунд, т.е. мгновенно).

7. В окне кода перетащите конструкцию **do in order**. В конструкцию перетащите нашу переменную aList, в результате чего появится диалоговое окно, как на рис. 160. Выберите опцию **insert <item> at end of world.my first method.aList** (добавить очередной элемент в список aList), далее последовательно выбирайте каждое транспортное средство. Всего у вас появится четыре вставки объектов в список.

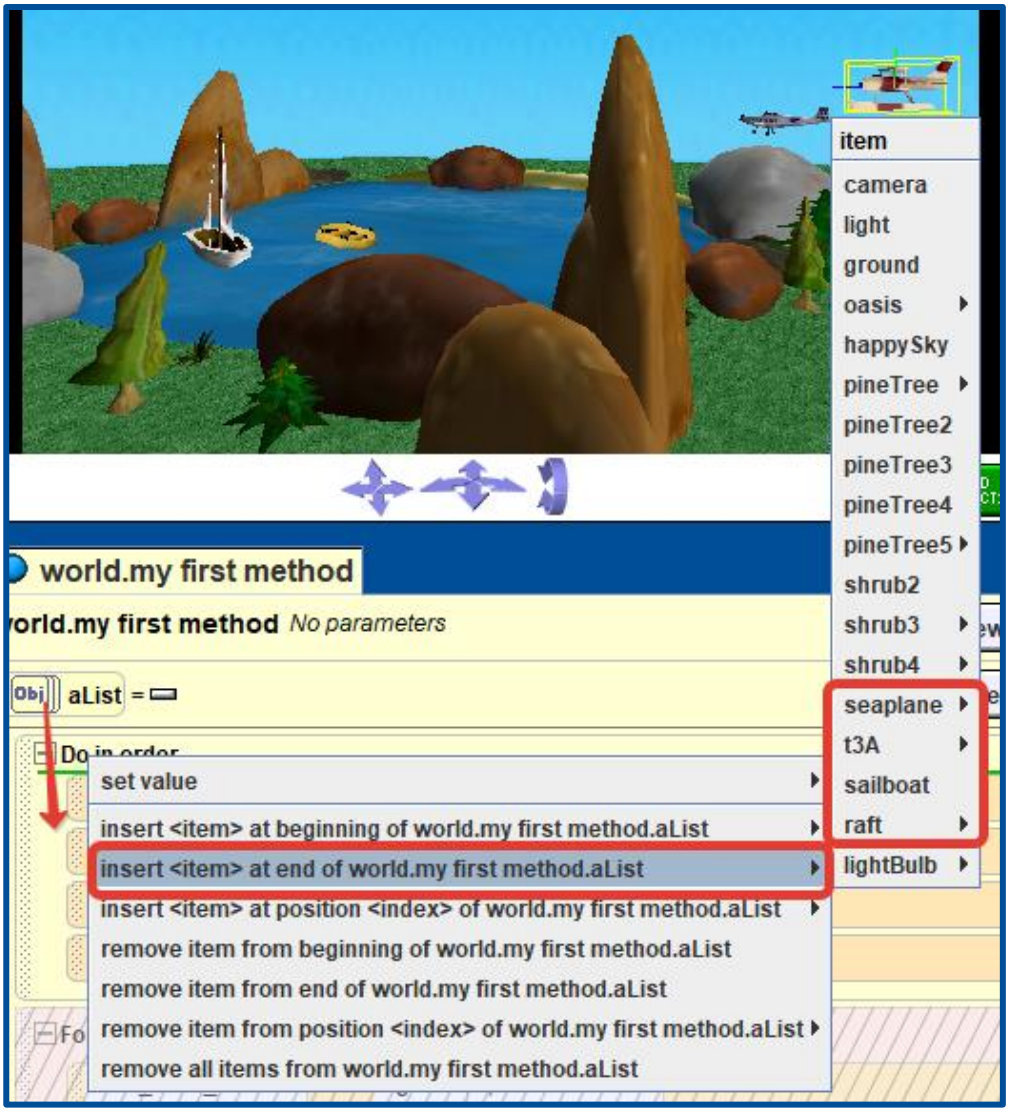


Рис. 160. Добавление элементов в список

8. Далее создадим алгоритм, в котором для каждого элемента списка по очереди происходит поворот направо на 0.2 оборота и движение вперед на 1 метр.

Перенесите конструкцию **for all in order** из нижней части окна кода в код. В появившемся при переносе списке выберите список aList.

В тело цикла **for all in order** перетащите объект **item_from_aList**

и в появившемся диалоговом окне укажите метод **turn right 0.2 revolution** (поворот направо на 0,2 оборота).

Затем сделайте так, чтобы все наши объекта двигались вперед на 1 метр: **item_from_aList move forward 1 meter**.

9. После разворота и начала движения всех транспортных средств сделайте так, чтобы объекты выполнили 5 циклов движения вперед одновременно. Для этого аналогично используйте цикл **for all together**, который будет вызываться из цикла **loop.** Результат алгоритма показан на рис. 161.

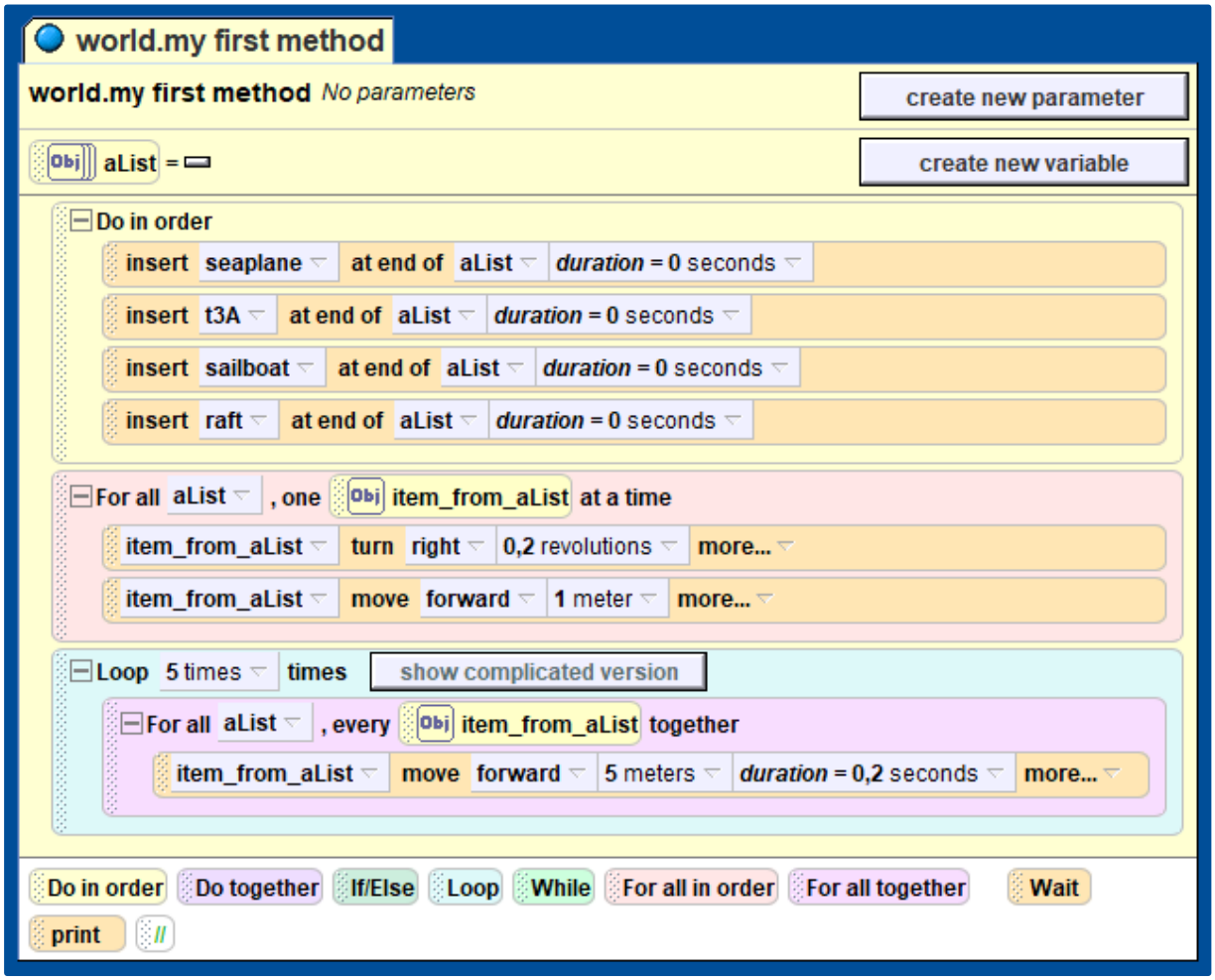


Рис. 161. Алгоритм движения лодок и самолетов

Самостоятельно сделайте так, чтобы лодки разворачивались и плыли назад (одновременно), а самолеты одновременно кружились над горами и возвращались назад. Для этого вам понадобится создать два дополнительных списка: для лодок и самолетов.

Задание 2. Волшебное появление пингвинов

Создадим проект, в котором, в зависимости от того какое число ввёл пользователь, будет появляться заданное количество пингвинов. Здесь мы работаем с новым свойством (property) – прозрачность (opacity).

1. Создайте новый мир на основе снега. В нём расположите 5 пингвинов (вкладка **Animals**). Для каждого пингвина на вкладке **properties** (свойства, окно свойств, методов и функций) установите значение свойство **opacity** (прозрачность), равным 0 (invisible) (рис. 162). Каждый пингвин в начале проекта будет невидимым.

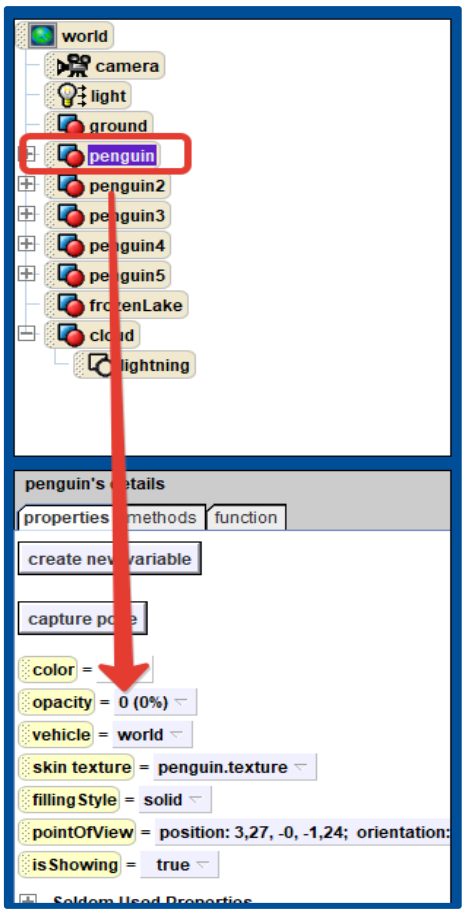


Рис. 162. Создание невидимого пингвина

2. Добавьте объект **Cloud** (облако). У этого объекта есть подобъект **lightning** (молния). Молнию в начале проекта также сделайте невидимой. Можно добавить другие объекты в окружение (замерзшее озеро, камни, скалы).

3. Добавьте переменную числового типа limit и значение по умолчанию у нее сделайте равной нулю.

4. Создайте список пингвинов. Назовите его **penguins**. Установите для переменной **penguins** значение объект и добавьте галочку **make a list**. С помощью кнопки **new item** добавьте всех пингвинов к этому списку (рис. 163). Это второй способ создания списка объектов, когда объекты в список добавляются не через код программы, а в результате создания списка.

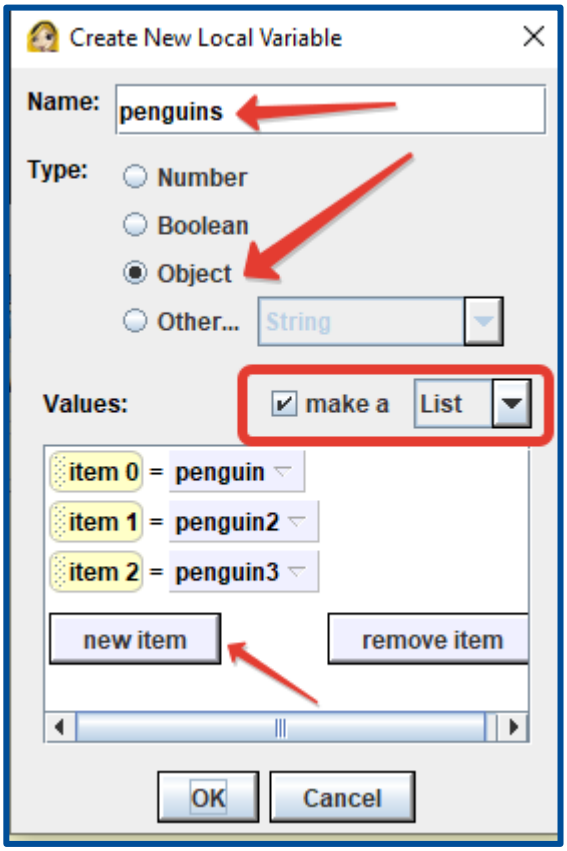


Рис. 163. Добавление объектов к списку вторым способом

5. В процессе написания алгоритма вам понадобится создать цикл **Loop**, который будет задавать какие-то действия для каждого пингвина из списка. Перенесите цикл **Loop** в редактор кода. В появившемся диалоговом окне выберите любое количество повторений цикла.

Нажмите на кнопку **show complicated version**

, чтобы изменить вид цикла. Установите конец цикла – переменную limit (рис. 164).

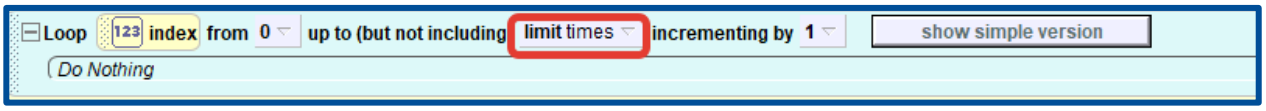


Рис. 164. Создание списка с параметром index

6. Перетащите в цикл **Loop** любого пингвина из дерева объектов. Выберите метод **set opacity** в значение **1**. В этом случае будет появляться только один пингвин.

Изменим это условие. Вместо указанного пингвина перетащите список **penguins**. В появившемся окне выберите **ith item from a list -> expressions -> index** (рис. 165).

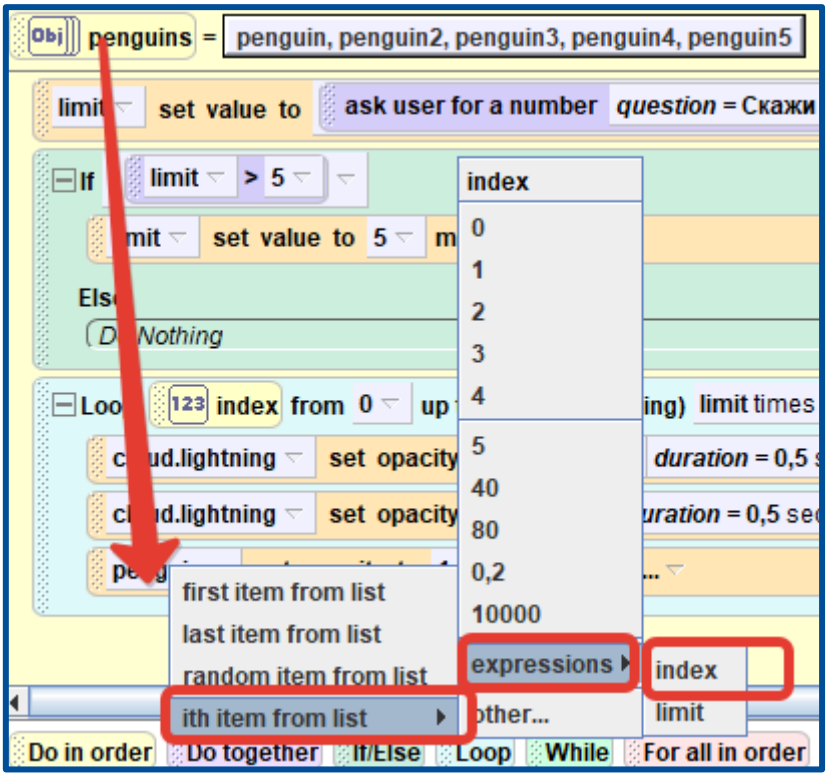


Рис. 165. Добавление индекса пингвинов к списку

7. Реализуйте весь алгоритм по следующему сценарию:

Переменные: (num) limit = 0, (object) penguins = [penguin, penguin2, penguin3, penguin4, penguin5].

Переменную limit сделать равным числу, которое вводит пользователь.

Если limit > 5, то

Limit сделай равным 5

Иначе ничего не делай

Цикл для index от 0 до limit с шагом по 1.

Молнию облака сделай видимой на 0,5 секунды.

Молнию облака сделай невидимой на 0,5 секунды.

Пингвина по номеру index сделай видимым.

Облако говорит: «Я исполнило твое желание!»

8. Дополните алгоритм: пингвины, которые были невидимыми, должны появляться, медленно меняя прозрачность. Затем они снова должны исчезать.

2.5. Лабораторная работа 5. Циклические алгоритмы в Alice

В этой лабораторной работе изучим основные отличия цикла с определенным количеством повторений **Loop** от цикла **While**. Мы создадим проект, в котором зайчик будет есть брокколи.

Цикл **Loop** выполняет простое зацикливание, когда вы знаете, сколько раз вы хотите повторить раздел кода.

Цикл **While** продолжает повторяться до тех пор, пока условие истинно, и останавливается, когда условие ложно.

Задание 1. Зайчик ест брокколи

1. Создайте новый мир Alice. Поместите в него зайчика (bunny) и несколько broccoli из папки Kitchen/Food. Чтобы брокколи не «висели в воздухе», кликните по брокколи правой кнопкой мыши и выберите **methods -> move to -> ground (переместить в землю).** Это поможет им «спуститься на землю». Затем нажмите на них и отодвиньте их все далеко за зайчика. Вы можете использовать кнопку «вверх» (↑) / «вниз» (↓), чтобы управлять объектами (рис. 166).

Рис. 166. Создание мира с зайцем и брокколи

2. Создайте новый метод **bunny.hop** – прыжок зайчика. Для этого на вкладке **methods** найдите кнопку **create new method**. Обратите внимание, что в дереве объектов должен быть выбран **bunny**. Назовите метод **hop**. Вы увидите, что создалась новая вкладка для написания кода.

3. Напишите код прыжка зайчика: последовательное выполнение движения вверх и вниз на 0,5 метра и одновременно с этим движение вперед на 0,5 метра. Чтобы это происходило быстро, добавьте из опции **more** значение **duration** равное 0,25.

Рис. 167. Код прыжка зайчика

4. Из главного метода **my first method** сделайте вызов метода **bunny.hop** и протестируйте его работу. Зайчик совершает прыжок ровно один раз.

5. Сделаем так, чтобы зайчик совершал прыжок 4 раза. Для этого перетащите **Loop** в окно кода **my first method**, выберите **other** и с помощью калькулятора укажите значение 4 (рис. 168). Запустите мир и посмотрите, сколько раз подпрыгнул зайчик.

Рис. 168. Повторение 4 раза

6. Теперь мы научим зайчика есть одну из брокколи. Мы не знаем, как далеко находится брокколи, поэтому нам нужно, чтобы зайчик несколько

раз перепрыгивал к брокколи, пока зайчик не окажется рядом с ней. Создайте новый метод **bunny.eat** аналогично, как это делали с методом **bunny.hop**. Добавьте в код метода **bunny.eat** строку (рис. 169).

Рис. 169. Зайчик поворачивается лицом к брокколи

7. Чтобы было удобно тестировать отдельные методы, можно запускать мир с тестируемого метода. В окне **Events** (события) в выпадающем списке **When the world starts** выберите нужный вам метод (рис. 160).

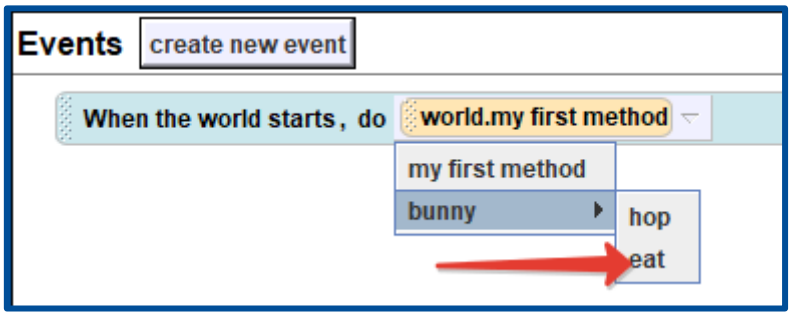


Рис. 170. Вызов метода bunny.eat в начале мира

Нажмите Play, чтобы увидеть, как зайчик поворачивается к брокколи.

8. Чтобы повторить код, основанный на условии, нам понадобится цикл **While**. Перетащите в код цикл **While** (он расположен снизу окна кода) и выберите **true** (рис. 171).

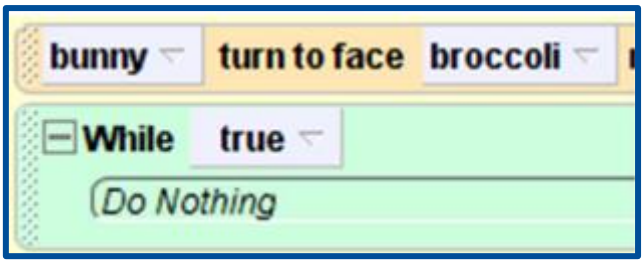


Рис. 171. Цикл while

Общая структура цикла while следующая:

while <условие>

<код>

Если условие истинно, код выполняется. Если условие по-прежнему выполняется, код выполняется снова. Это повторяется до тех пор, пока условие не станет ложным, после чего код в цикле больше не выполняется.

9. Мы хотим, чтобы зайчик продолжал прыгать к брокколи до тех пор, пока расстояние между ними будет больше 0,6. Нажмите на **world** в дереве объектов, затем на вкладку **functions** (функции).

Перетащите **a>b** в поле **true** вашего цикла **While**, выберите 1 и 0.6 (рис. 172).

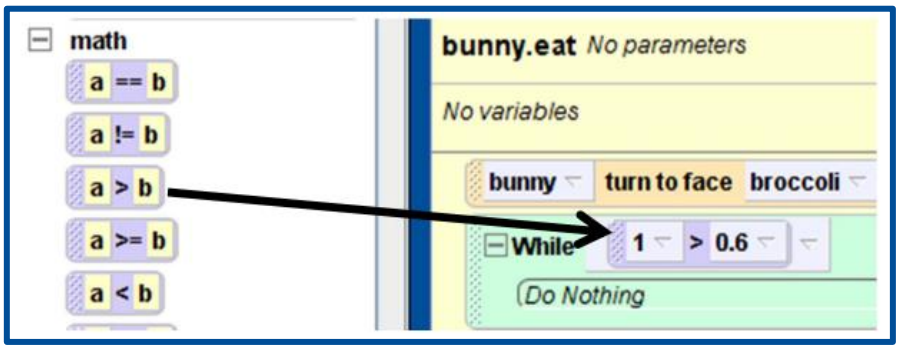


Рис. 172. Добавление логического условия для цикла While

Нажмите на объект **bunny** в дереве объектов и перейдите на вкладку **functions**. Перетащите функцию **bunny distance to** (расстояние от зайчика до) на 1 и выберите брокколи (рис. 173).

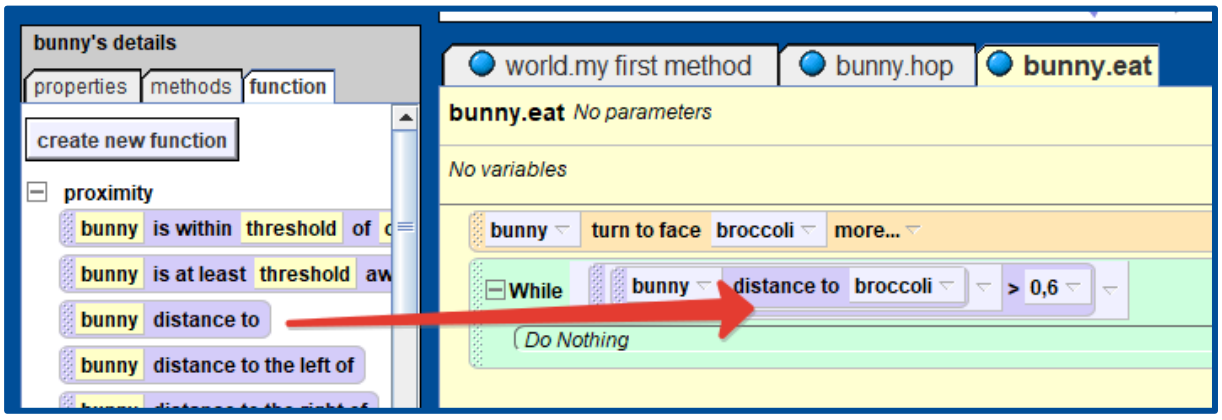


Рис. 173. Расстояние от зайчика до брокколи

В дереве объектов выберите **bunny**, на вкладке **methods** и перетащите метод **bunny.hop** в раздел **Do nothing**.

Нажмите на **Play** (запустите мир) и посмотрите, как зайчик направляется к брокколи и останавливается, когда находится ближе чем 0,6 метра.

10. Теперь добавьте несколько команд, чтобы зайчик наклонялся и съедал брокколи. Добавьте после цикла **While** команду **0.2 revolutions**, чтобы **bunny** повернулся вперед 0.2, установите свойство **isShowing** равным **false** чтобы брокколи исчезла, а зайчик повернулся назад (рис. 174).

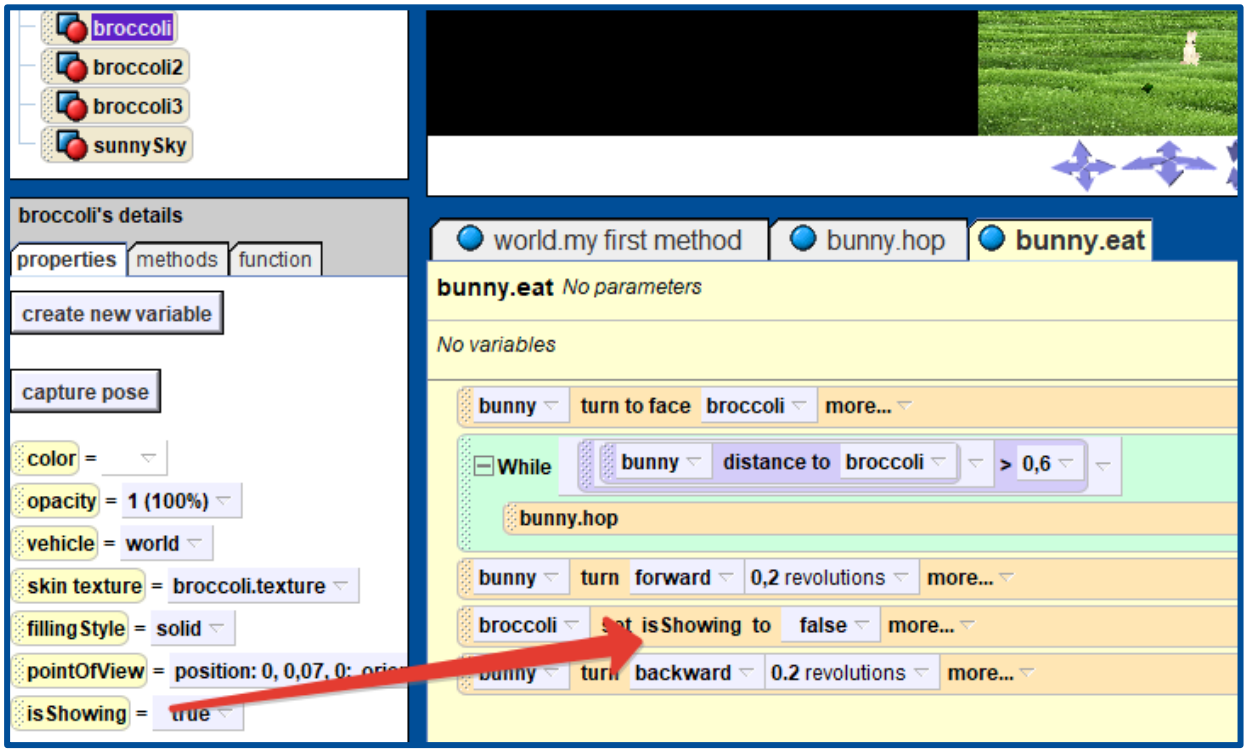


Рис. 174. Код для метода bunny.eat

Нажмите кнопку **Play** и наблюдайте, как зайчик ест брокколи.

11. Нам нужно настроить параметры брокколи, чтобы зайчик мог есть другие кусочки брокколи. В методе **bunny.eat** нажмите **create new parameter**, назовите его **tastyMorsel** (лакомый кусочек) и выберите **object** для типа параметра. Перетащите **tastyMorsel** поверх всех трех вариантов брокколи (рис. 175).

12. Теперь закончите рассказ. Во-первых, измените событие **When the world starts** обратно на **my first method** (рис. 170).

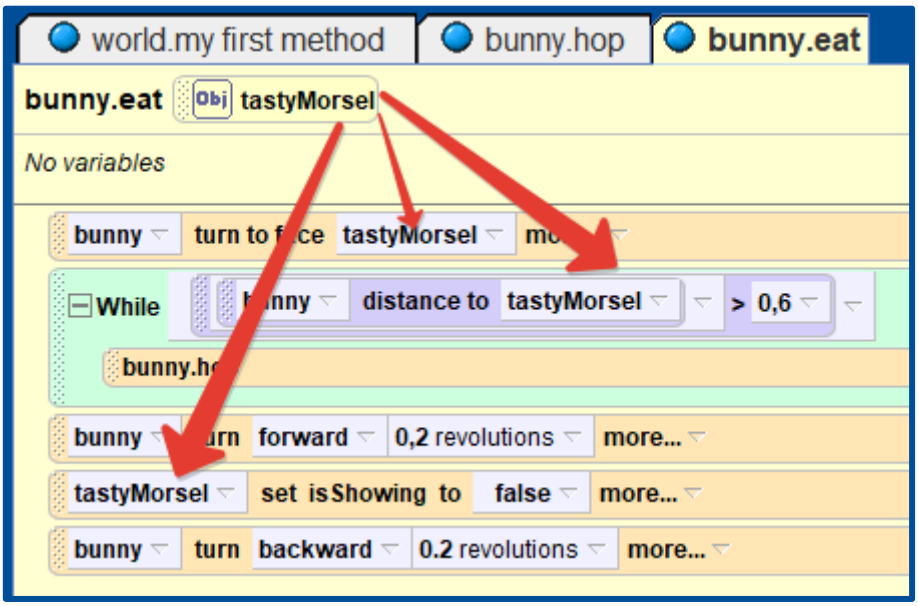


Рис. 175. Замена параметра broccoli на tastyMorsel

13. Перейдите на вкладку **world. My first method**. После **While** введите конструкцию **do in order**. Из вкладки **methods** для объекта **bunny** трижды перетащите метод **eat**. В качестве **tastyMorsel** выбирайте каждый раз очередную брокколи. Окончательный код для **my first method** показан на рисунке 176.

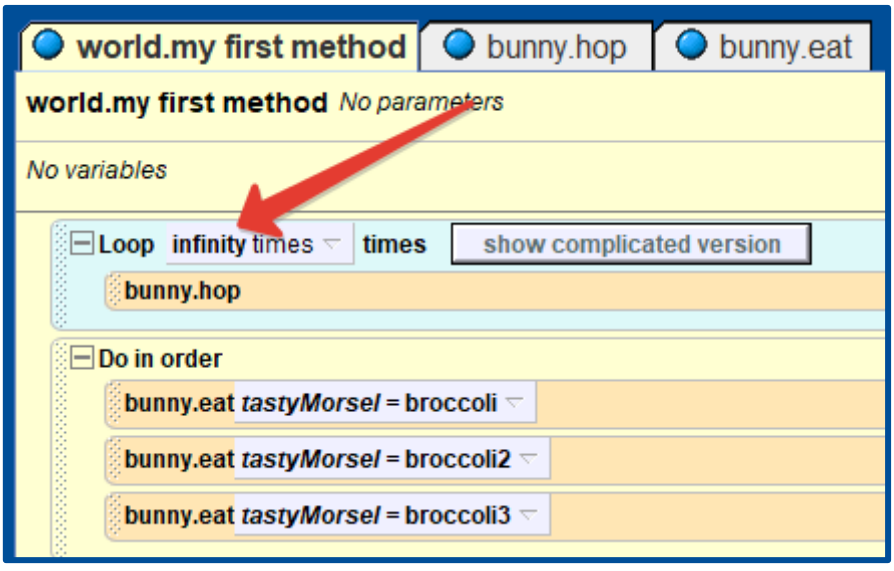


Рис. 176. Бесконечное выполнение цикла

14. Нажмите на **Play**, чтобы посмотреть, как зайчик съест всю брокколи.

15. Вместо количество повторений 4 раза, нажмите на **infinity** (выберите бесконечное количество раз. Снова запустите мир. Когда зайчик съест брокколи? Никогда! Бесконечность опасна в использовании. Действия после условия **infinity** никогда не произойдут.

16. Измените значение **infinity** обратно на **4**.

Задание 2. Добавьте список food и поместите в него 3 уже имеющиеся брокколи, добавьте в сцену еще и carrot (морковку, папка Kitchen/Food). Для объекта **World** на вкладке **properties** нажмите на кнопку **create new variable**. Создайте переменную **food**, тип **Object**, поставьте галочку на **make a list** (список). К списку добавьте все брокколи и все морковки (рис. 177).

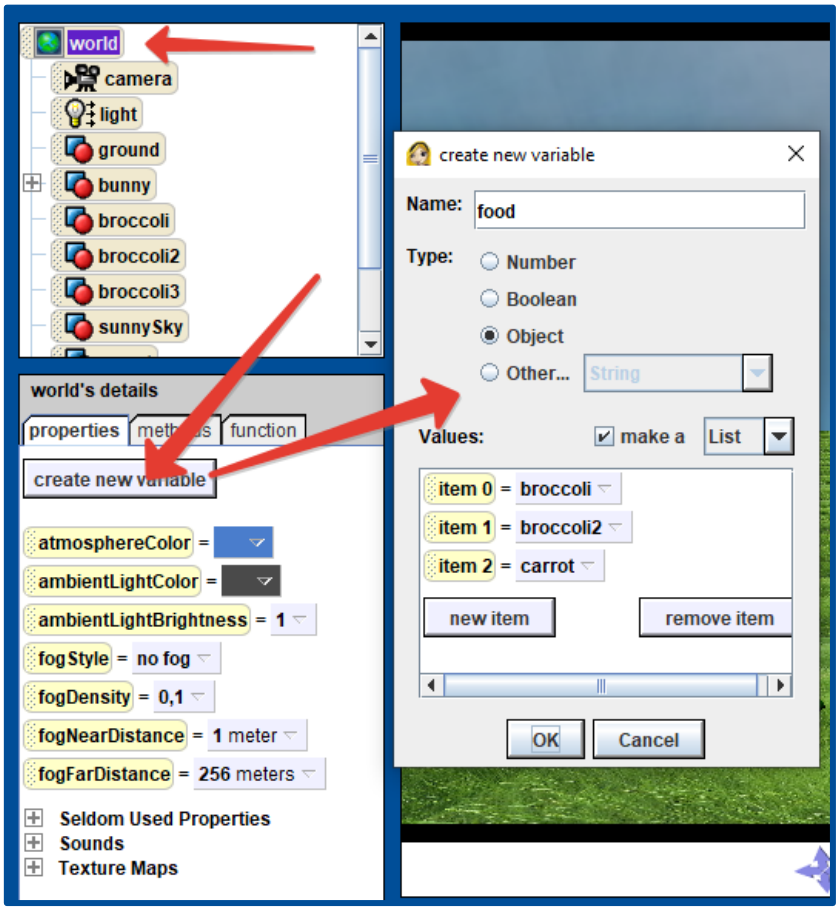


Рис. 177. Создание списка food

В коде **my first method** отключите или удалите опцию **Do in order**. Перетащите снизу вверх раздел **Do all in order**, выберите **expressions -> world.food**. Нажмите на объект bunny, перетащите метод bunny.eat в тело цикла for all in order и выберите expressions -> **item_from_food** (рис. 178).

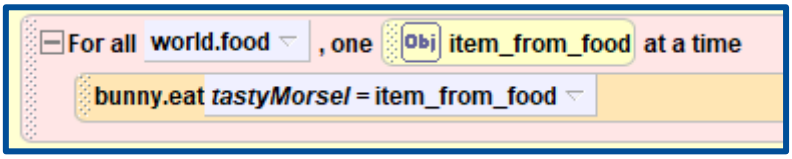


Рис. 178. Цикл для каждого объекта из списка

Запустите мир, чтобы посмотреть, как зайчик ест еду.

Задание 3. Самостоятельно создайте мир, в котором лягушонок будет прыгать от камушка к камушку и сообщать, сколько ему пришлось сделать прыжков до очередного камушка. Камушки можно представить в виде списка. Для подсчета количества прыжков нужно завести дополнительную переменную.

2.6. Лабораторная работа 6. Управление освещением в Alice

Создадим проект, в котором представим ситуацию с падением метеорита на планету Земля 66 млн лет назад. Как известно, падение метеорита послужило причиной гибели всех динозавров.

В нашем проекте отработаем несколько новых навыков:

- работа с освещением;
- управление камерой через фиктивные объекты;
- управление текстурами;
- методы для работы с освещением;
- метод Turn to face (повернуться «лицом» к объекту);
- Метод Move to (двигаться к объекту).

У методов поворота (turn, roll), изменения размера (resize) и некоторых других дополнительно в опции **more** можно добавить **as seen by** (как видно из). Рассмотрим, как работает эта опция.

Мы будем использовать объекты «планета Земля», «камера» и ground в мире Алисы, чтобы продемонстрировать **as seen by**. Поместите метод **turn** в свой редактор кода, чтобы дать команду камере повернуть направо на один оборот (**turn tight 1 revolution**). Укажите в качестве объекта, вокруг которого будет происходить вращение, планету Земля. Камера будет совершать 1 оборот вокруг объекта «планета Земля». На рис. 179 показана траектория движения камеры.

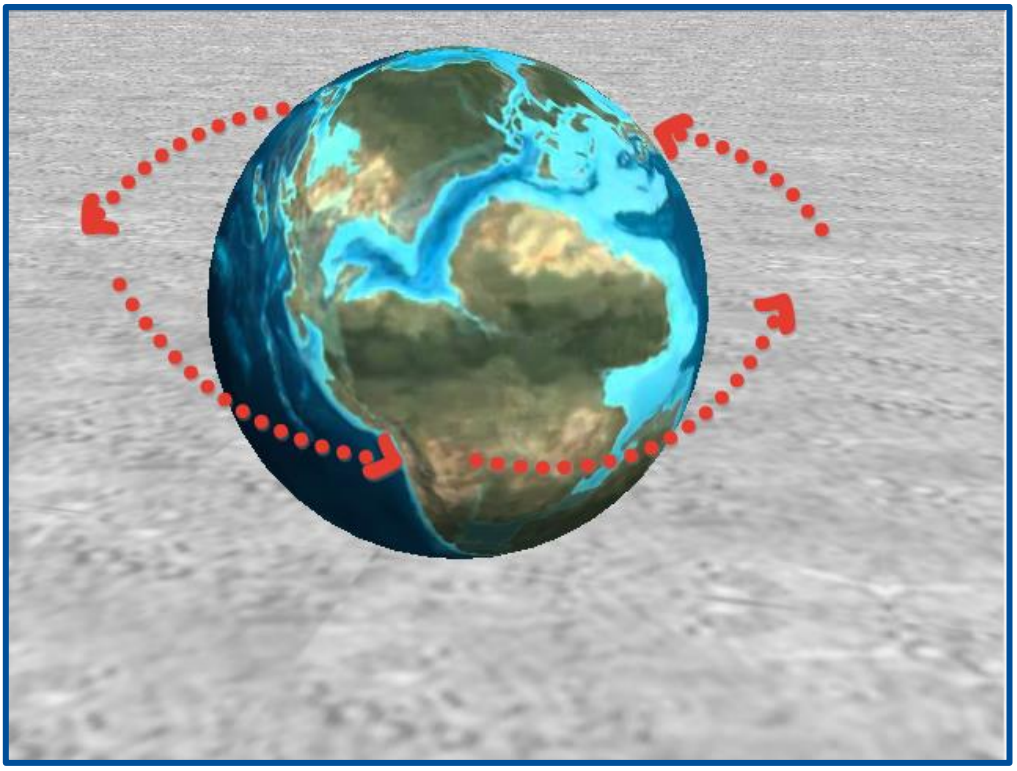


Рис. 179. Вращение вокруг объекта планета Земля

Если же вместо камеры мы укажем объект **ground**, то поверхность уже будет вращаться вокруг объекта «планета Земля».

Таким образом, опция **as seen by** задает объект, вокруг которого происходит действие.

Чтобы источник освещения при вращении создавал эффект как при восходе и закате Солнца, можно также применить метод вращения с дополнением as seen by. В этом случае объектом вращения выступает, например, **light** (лампа), а в качестве объекта **as seen by** выбирается планета.

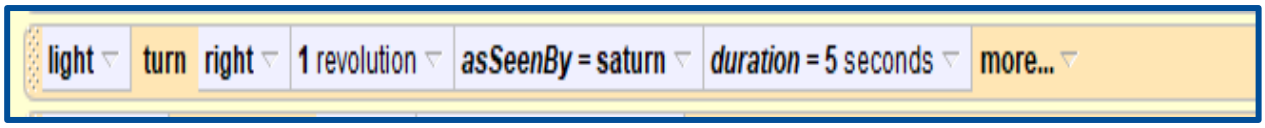


Рис. 180. Источник света будет вращаться вокруг Сатурна

Объектом **light** (лампочка), который по умолчанию присутствует в каждом проекте, можно управлять через некоторые специфические методы.

Turn to face (поворот «лицом») включает в себя вращение так, чтобы центральная точка объекта была направлена на цель.

Point at включает в себя как вращение, так и движение вперед/назад, так что центральная ось объекта образует прямую линию с целью.

Хотя эти методы сильно похожи по своему действию, разница между ними становится более очевидной, если вы пытаетесь повернуться «лицом» к целям выше или ниже объекта. Попробуйте эти методы на простых объектах, чтобы понять, в чем разница.

У объекта **light** (лампа) есть несколько базовых свойств (они представлены на вкладке **properties**):

- **color** – для изменения установки цвета лампы;
- **brightness** – для изменения установки свойств яркости;
- **range** – уровень для изменения дальности освещения.

В процессе создания своего мира вы можете в коде программы менять значения этих свойств. Например, источник освещения может сменить цвет с белого на оранжевый и стать более ярким (рис. 181).

Рис. 181. Программное изменение цвета и яркости источника освещения

Вы можете установить в мир Alice дополнительные источники освещения. Их можно найти, нажав кнопку **Add Objects**, в папке **Lights.** Здесь представлены источники освещения: **blueLight** (синяя лампа), **greenLight** (зелёная лампа), **redLight** (красная лампа), l**ightBulb** (обычная лампочка), **spotLight** (прожектор) и **directionalLight** (направленный свет). Вы можете установить несколько источников освещения для лучших эффектов и управлять ими.

Интересный эффект получается, если заставить лампочку поворачиваться на скорости вокруг какого-нибудь объекта. Для источников освещения в группе методов на вкладке **methods** существуют методы, которые выполняются на скорости. К ним относят:

- **move at speed** (двигаться на скорости);
- **turn at speed** (поворачиваться на скорости);
- **roll at speed** (вращаться на скорости).

Как можно достичь чередования красного и зеленого освещения объекта при вращении источников света, показано на рис. 182–183.

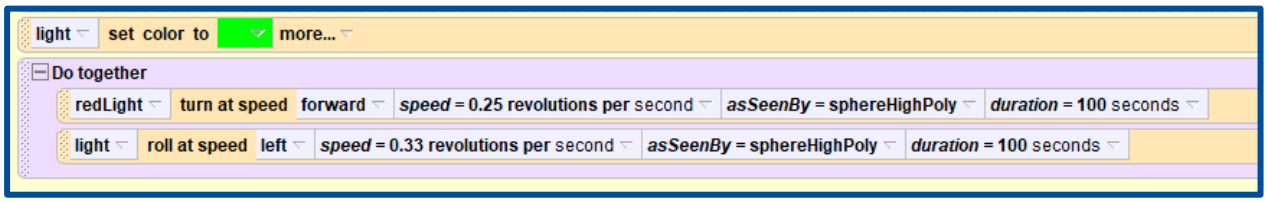


Рис. 182. Одновременное вращение двух источников света

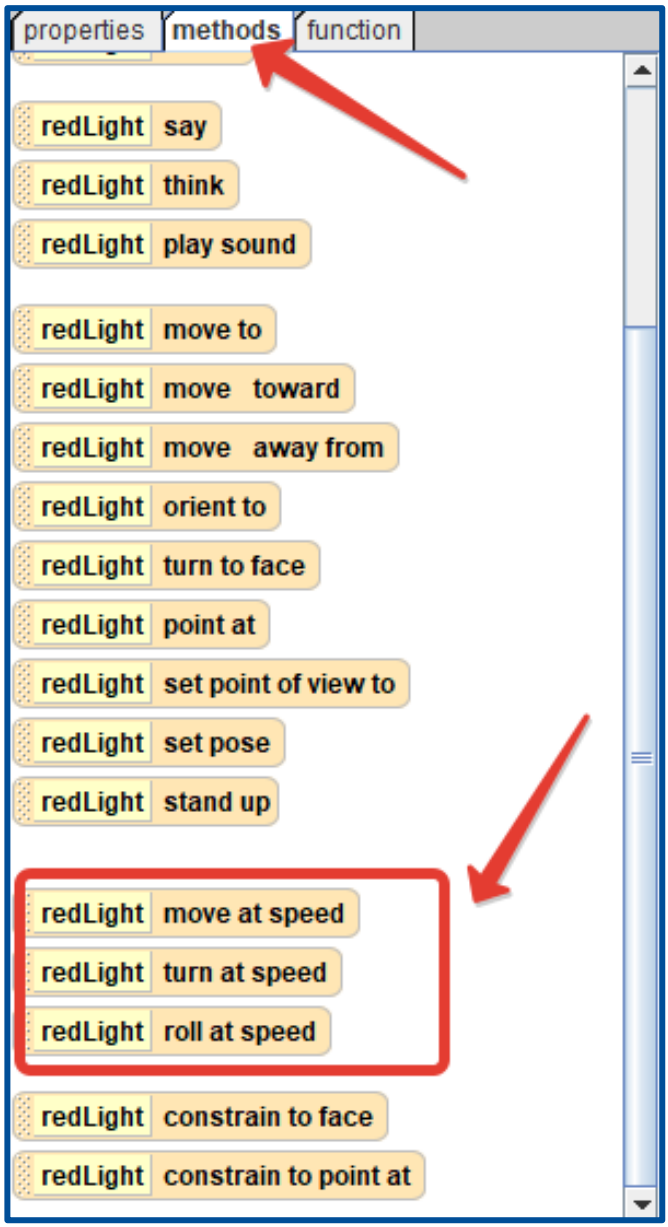


Рис. 183. Методы для управления объектом на скорости

Управление камерой через фиктивные объекты

Метод **set point of view** (установить точку обзора) применяется для камеры, чтобы можно было расположить ее в удачном месте.

Dummy объект (фиктивный) – это объект для управления камерой, похож на штатив для камеры. Она сохраняет местоположение вашей камеры. Таким образом, если вы перемещаете камеру, вы всегда можете вернуться в определенное положение, переместившись в фиктивное местоположение камеры.

Посмотрите в правую часть экрана и найдите серую кнопку под кнопками позиционирования объектов с надписью **more controls** (дополнительные элементы управления) (рис. 184).

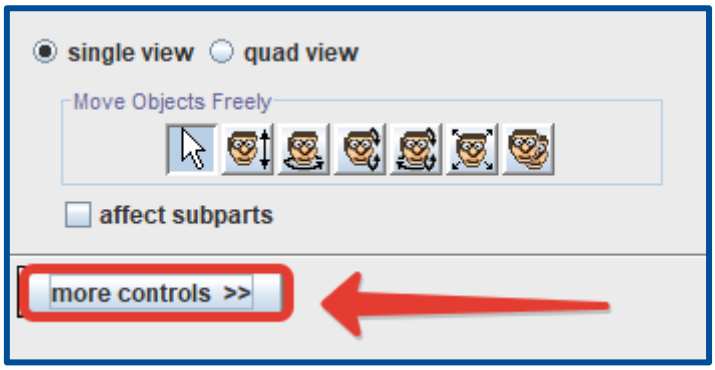


Рис. 184. Дополнительные элементы управления

Дополнительные кнопки появятся после того, как вы нажмете **more controls** (дополнительные элементы управления).

Нажмите на кнопку **drop dummy at camera** (установить манекен на камеру) (рис. 185).

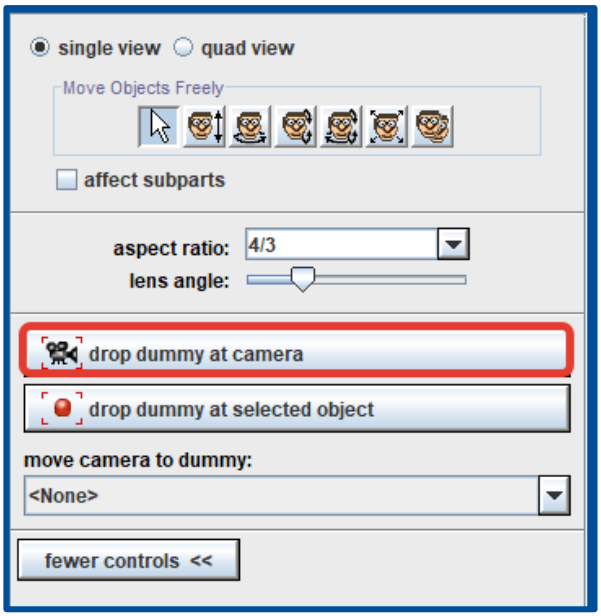


Рис. 185. Установка позиции камеры на манекен

Как только вы нажмете эту кнопку, в вашем дереве объектов появится папка с надписью **Dummy Objects** (фиктивные объекты). Если вы нажмете

на знак плюс рядом с манекеном, то в папке **Objects** появится список ваших фиктивных положений камеры (рис. 186).

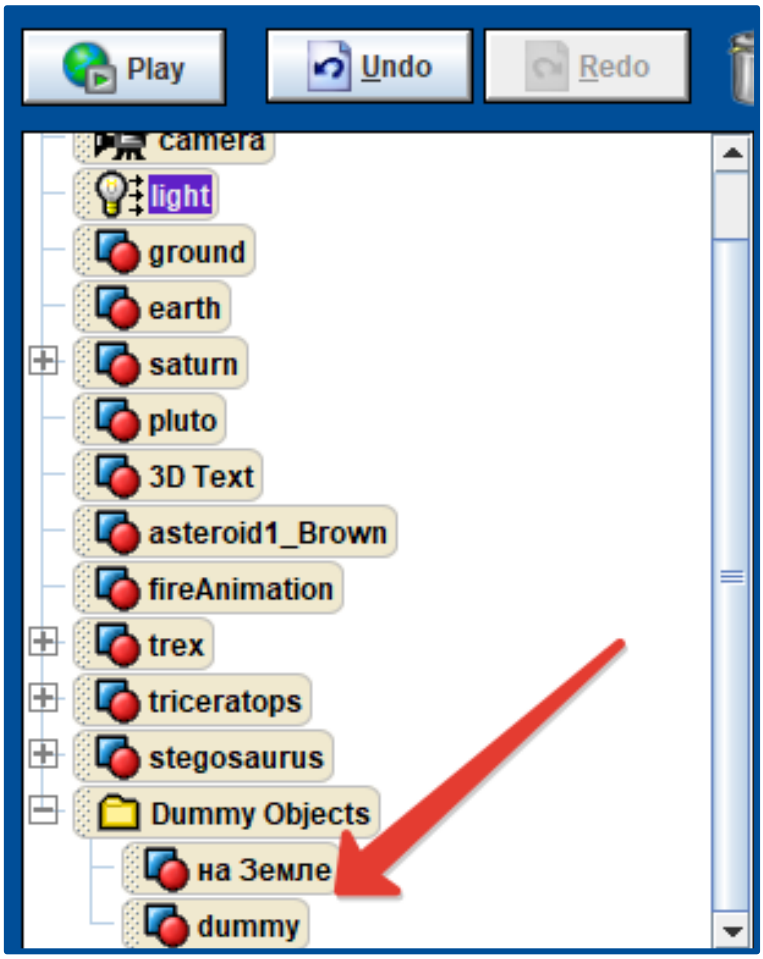


Рис. 186. Список объектов манекенов

Всякий раз, когда вы добавляете фиктивное положение камеры, вам следует переименовать его, чтобы вы знали, в каком положении оно находится. Щелкните правой кнопкой мыши на **dummy** в дереве объектов, а затем выберите **rename** (переименовать).

Лучше всего добавлять манекен в исходное положение камеры всякий раз, когда начинаете новый мир Alice.

Теперь, когда у нас установлена фиктивная камера, мы можем свободно перемещать камеру, не теряя своего места. Под сценой есть три набора стрелок, которые перемещают камеру (рис. 187).

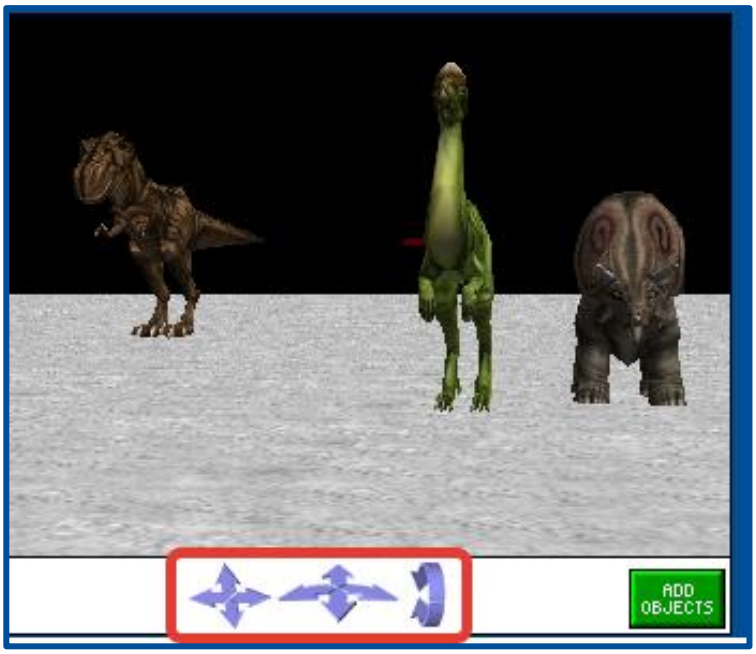


Рис. 187. Кнопки управления камерой

Первый набор перемещает камеру вверх, вниз, влево и вправо. Второй набор перемещает ее вперед и назад, и перемещает слева направо. При последнем наборе камера поворачивается вверх и вниз.

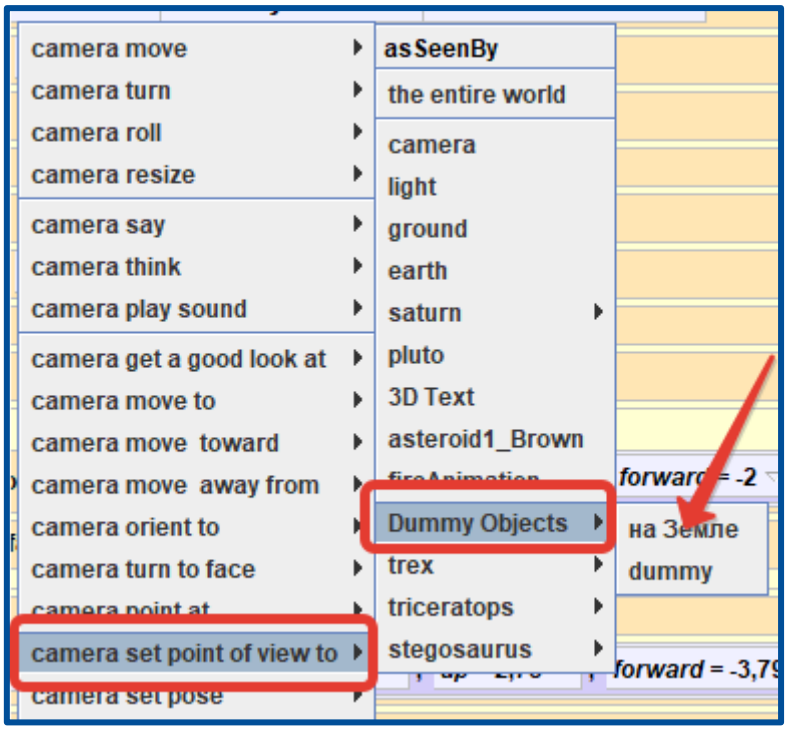


Рис. 188. Кнопки управления камерой

Чтобы переместить камеру влево нажмите и удерживайте стрелку влево во второй группе пока объекты не исчезнут из кадра. При перемещении мыши дальше от стрелок камера перемещается быстрее. Следите за тем, чтобы камера не перемещалась слишком далеко вверх или вниз.

Переместите объект (например, стегозавра) с помощью позиционных кнопок так, чтобы он располагался «лицом» к вам (камере). Нажмите **more controls** (дополнительные элементы управления, в режиме **Add Objects**), затем кликните **drop dummy at camera** (наведите манекен на камеру), чтобы сохранить этот вид. Переименуйте эту фиктивную камеру в название «На Земле».

Для управления движением камеры щелкните правой кнопкой мыши на камере в дереве объектов (можно попробовать из дерева объектов перетащить камеру в окно редактора кода, вы получите также список методов). В появившемся меню выберите **methods** (методы), среди методов выберите **camera set point of view to** (камера установить точку обзора на). Затем в списке выберите папку с фиктивными объектами (возможно Dummy Objects у вас переименовано).

Камеру можно направить и на конкретные координаты в мире. Пусть мы используем метод **move to** (движение к) для движения камеры. В дополнительной опции **more** можно выбрать **position.** В опцию добавляется контейнер для установки координат смещения объекта. Чтобы задать координаты, нужно в дереве объектов выбрать **world**, перейти на вкладку **functions** (функции) и выбрать right, up, forward. В этих координатах уже можно задать соответствующие значения справа, сверху и спереди.

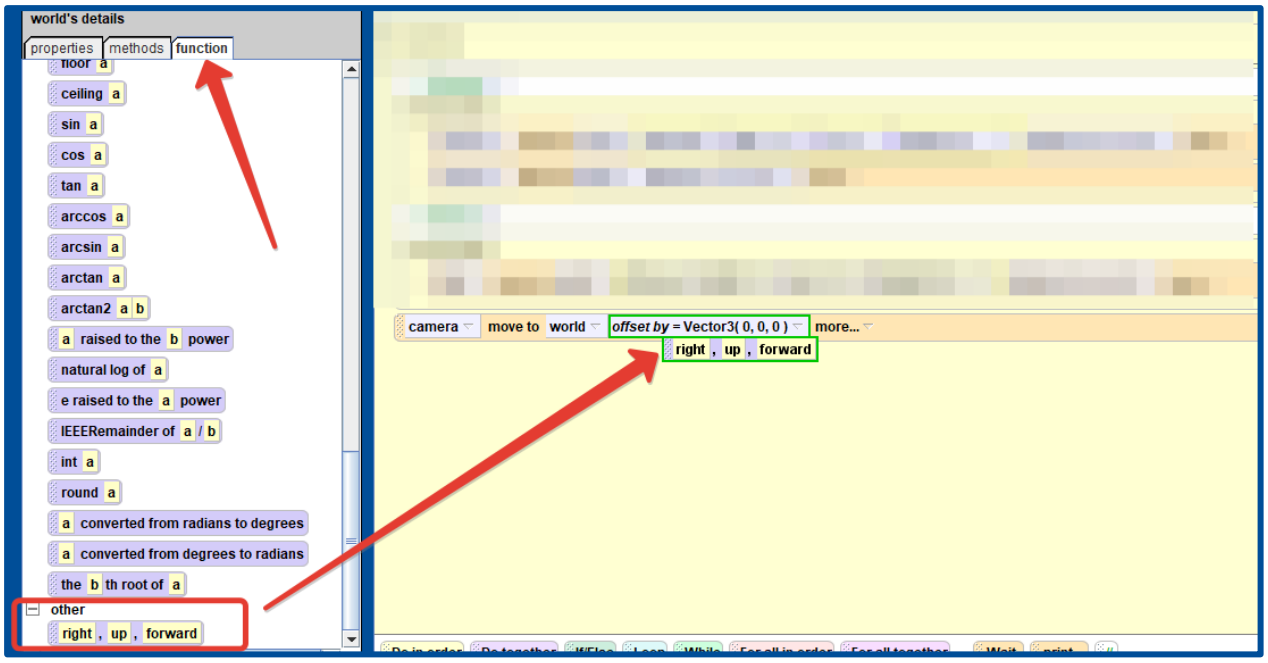


Рис. 189. Установка координат позиционирования

Управление текстурами

У объектов в Alice есть возможность управления текстурами. Например, вы можете заменить имеющуюся текстуру объекта на какую-нибудь другую. Для нашего будущего проекта объект «планета Земля» имеет не совсем подходящую текстуру (рис. 190).

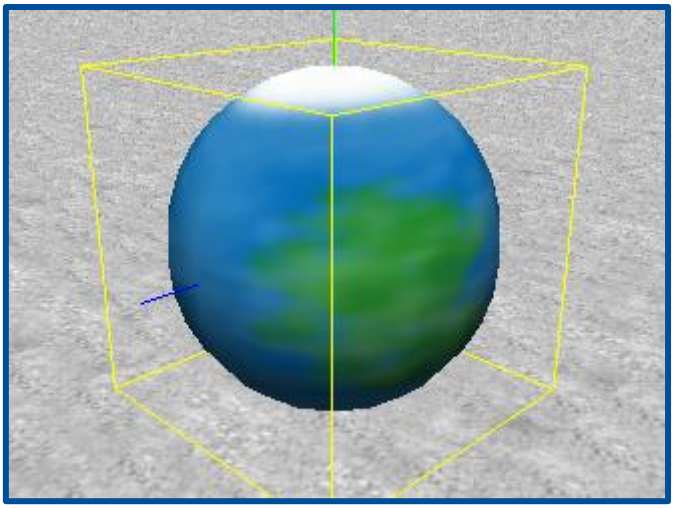


Рис. 190. Первоначальная текстура объекта планета Земля

Для нашей истории нужно подобрать текстуру, какой она была у планеты в момент падения астероида, уничтожившего динозавров примерно 66 млн лет назад. Попробуйте найти подходящее растровое изображение, например, как на рис. 191.

Рис. 191. Новая текстура объекта «планета Земля»

Новую текстуру нужно импортировать для установки. Нажмите на **File** (файл), затем **Import** (импорт). Найдите папку, в которую вы сохранили свою текстуру. Откройте ее, нажмите на свою текстуру. Затем нажмите кнопку **Import** (импорт). Когда вы нажмете кнопку **Import**, это будет выглядеть так, как будто в вашем мире ничего не произошло. Однако изображение с картинкой можно найти у объекта **World** на вкладке **Properties** (свойства) в разделе **Texture Maps** (рис. 192). Также текстуру можно «натянуть» на объект, например, на планету Земля или на сферу (качество будет лучше) и установить данную текстуру в свойстве **skin texture**.

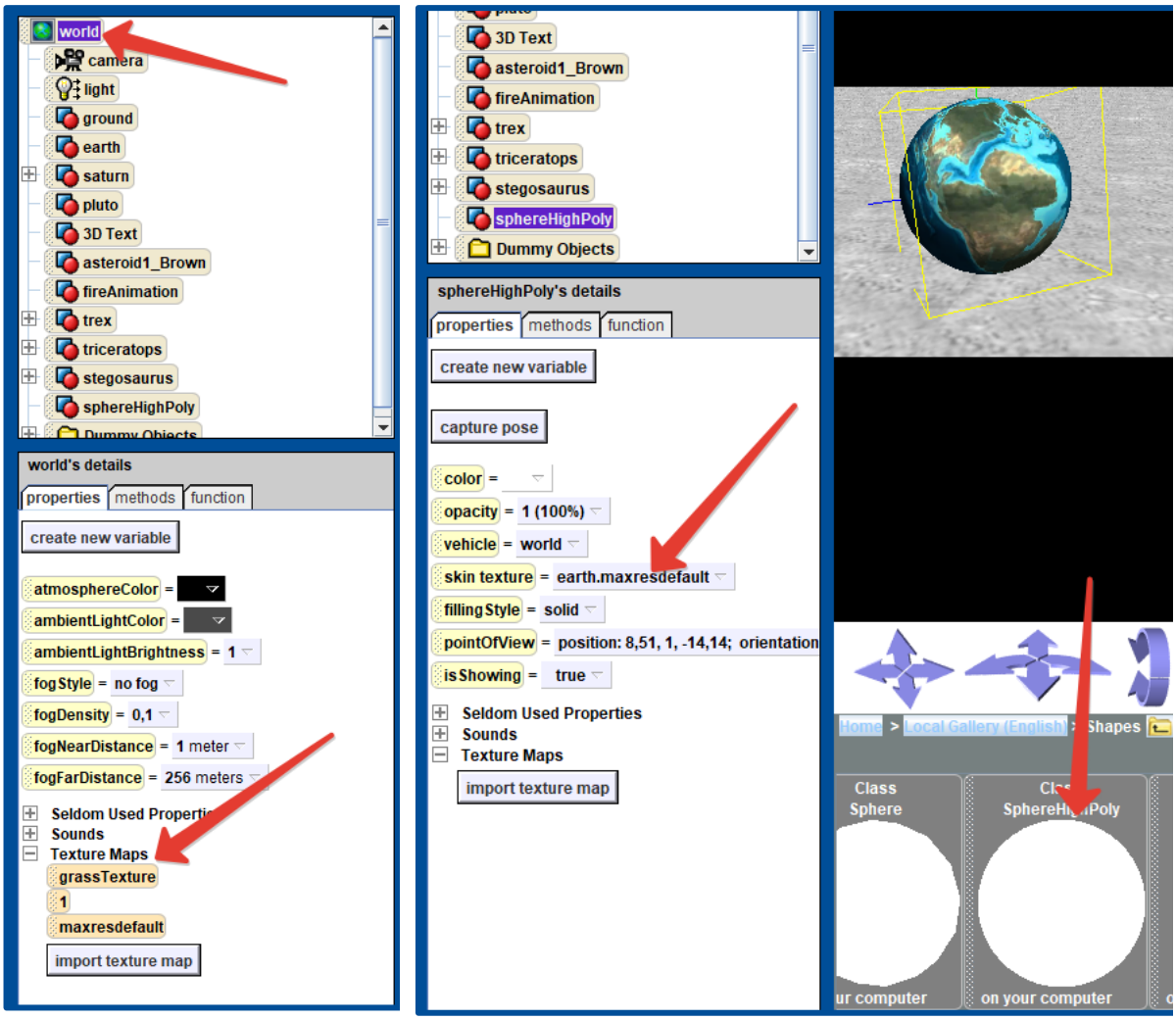


Рис. 192. Наложение текстуры на объект высокополигональную сферу

В процессе запуска вашего мира в Alice вы можете менять текстуру объектов. Для этого нужно перетащить свойство **skin texture** в окно кода. Появится метод **set skin texture to** (установить текстуру на), в котором можно выбрать нужную текстуру из списка импортированных. Например, если изначально вы создали мир на основе «лунной поверхности» (**moon**), то в процессе запуска мира вы можете сменить ее на «траву» (**grass**) (рис. 193).

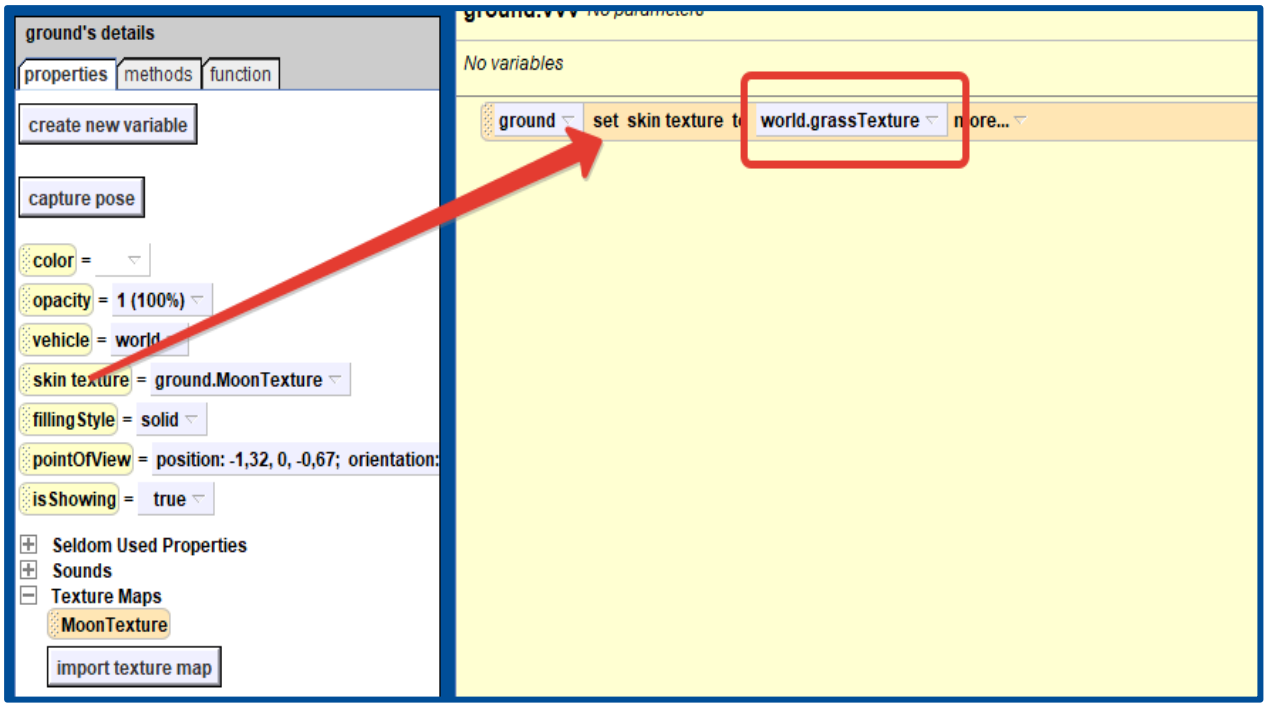


Рис. 193. Изменение текстуры при выполнении алгоритма

Задание. Падение астероида на Землю 66 млн лет назад. Гибель динозавров

1. Создайте новый проект и разместите, как это делали обычно, объекты – планеты солнечной системы (Земля, Сатурн, Юпитер, Нептун и др.).

2. Добавьте недалеко от планеты Земля объект **Asteroid**.

3. Добавьте объект **fireAnimation**, который будет имитировать огонь, когда астероид упадет на Землю.

4. Добавьте к сцене динозавров, расположите их на объекте **ground**.

5. Установите фиктивные объекты и настройте их на показ всех важных локаций камеры.

6. Создайте новый метод **world.setTheStage**, при котором камера подлетает к Сатурну. Затем источник освещения поворачивается вокруг Сатурна.

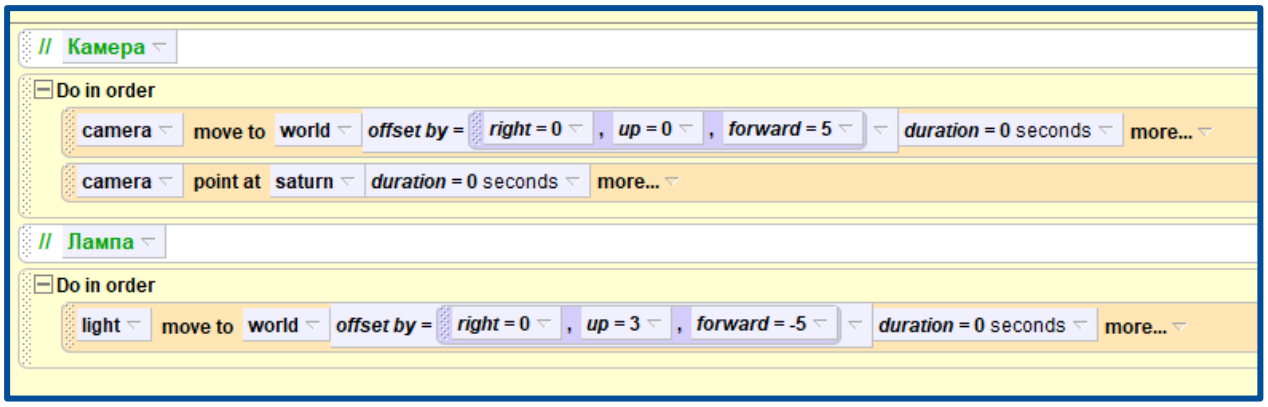
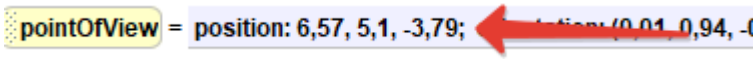


Рис. 194. Код облета Сатурна

7. Создайте самостоятельно аналогичные методы подлета камеры к другим планетам и их освещения лампочками. В качестве координат для лучшего вида камеры можно использовать как фиктивные объекты, так и координаты объектов, узнать которые можно в свойстве



(рис. 195).

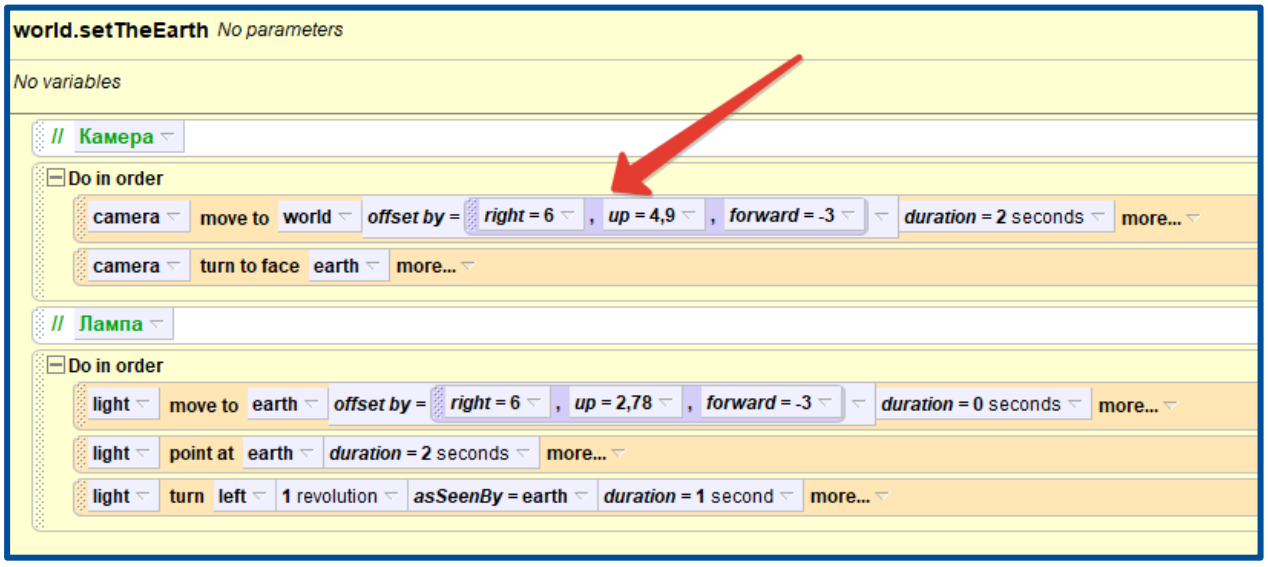


Рис. 195. Код облета Земли

8. Между методами можно добавлять wait (ожидание) 1 или 2 секунды.

9. Для тщательной отладки отдельных методов вызывайте их с помощью события **When the world starts** (рис. 196).

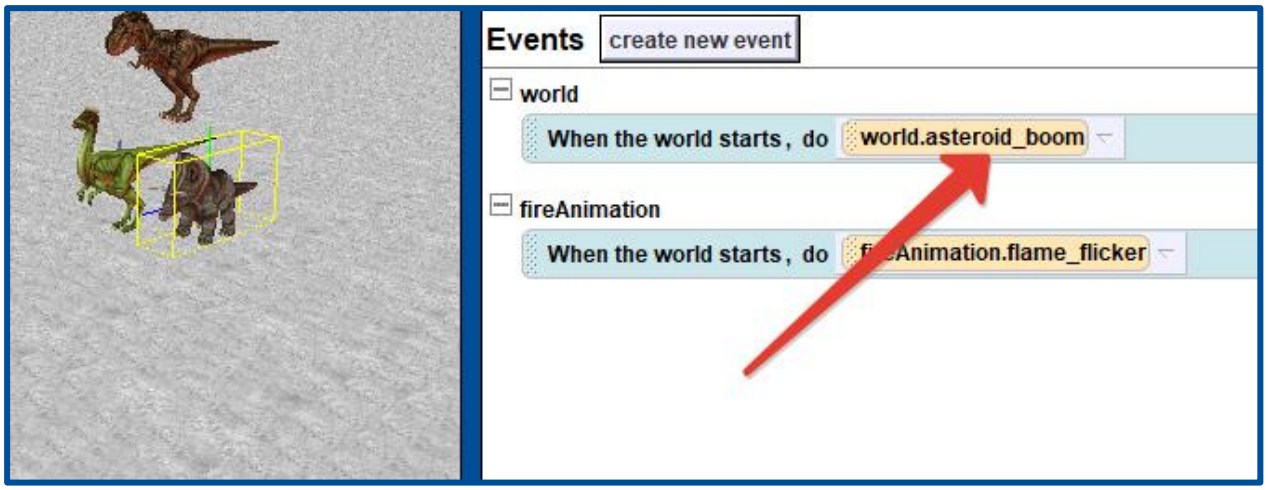


Рис. 196. Вызов метода world.asteroid_boom в событии when the world starts

10. Примерная раскадровка сюжета показана на рис. 197.

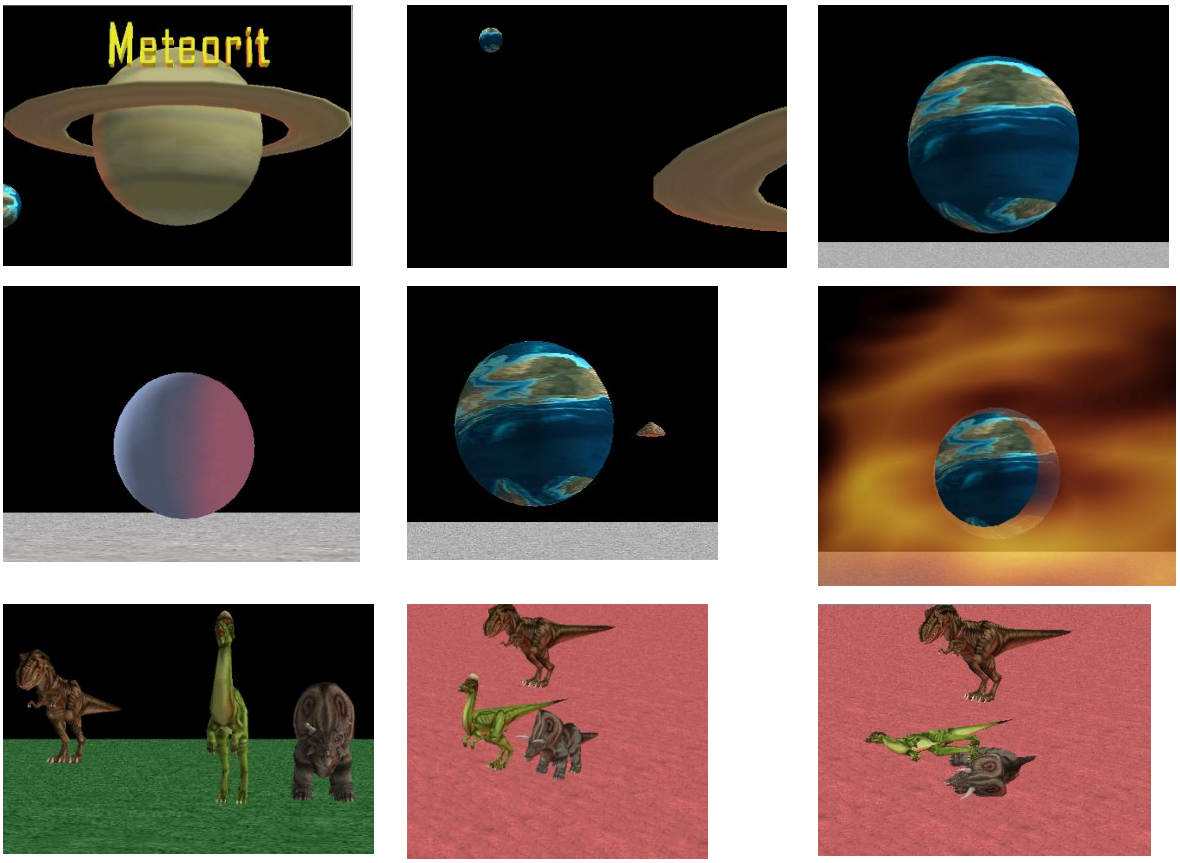


Рис. 197. Примерная раскадровка сюжета

1. Примерный алгоритм сюжета показан на рис. 198 (у вас могут быть другие координаты объектов).

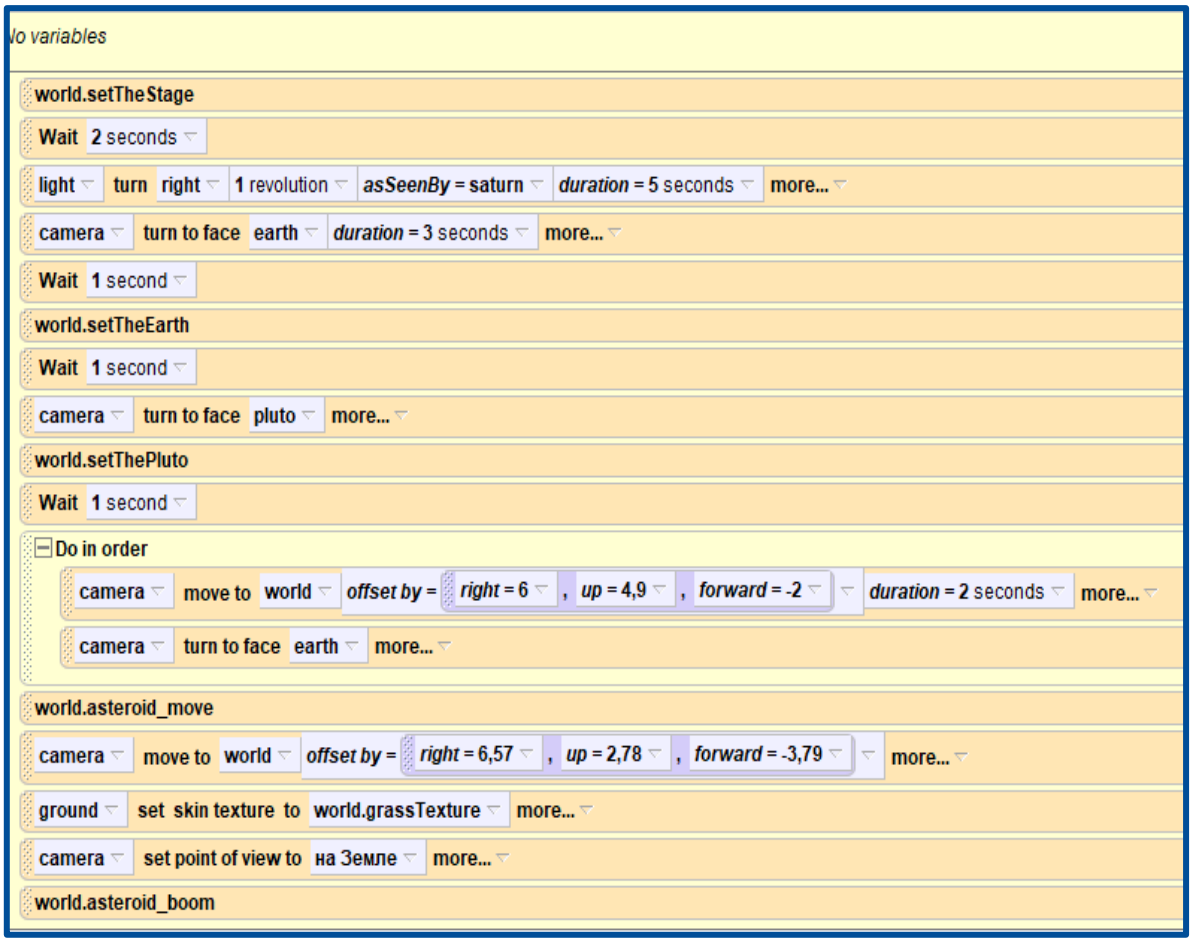


Рис. 198. Примерный алгоритм сюжета

2.7. Лабораторная работа 7. Математические алгоритмы в Alice

В этой лабораторной работе мы рассмотрим несколько задач с математическими алгоритмами, будем работать с арифметическими действиями (сложением, вычитанием, умножением и делением). Еще раз используем функции преобразования числа из вещественного в целочисленный тип, преобразование из числового типа в строку. Познакомимся с объектом 3D-текста. Научимся генерировать случайные числа (random). Закрепим работу с циклами и создание новых методов и свойств.

Задание 1. Взрыв вулкана

Создадим проект, в котором у нас будет 3D-текст, показывающий сколько секунд осталось до взрыва вулкана. Вы можете придумать небольшой сюжет, при котором, после того как таймер дойдёт до 0, будет происходить какое-то событие. Например, можно разместить вулкан, который будет взрываться, когда таймер завершит свой счёт. 10, 9, 8…

1. Создайте новый мир и разместите в нем вулкан, какой-нибудь остров, деревья на этом острове. У вулкана можно скрыть камни и лаву. Для этого в свойствах выберите для каждого подобъекта значение **isShowing** значение **false**. Разместите 3D-текст (**3D Text**), который в дереве объектов переименуйте в **timer**. Примерное расположение объектов показано на рисунке 199.

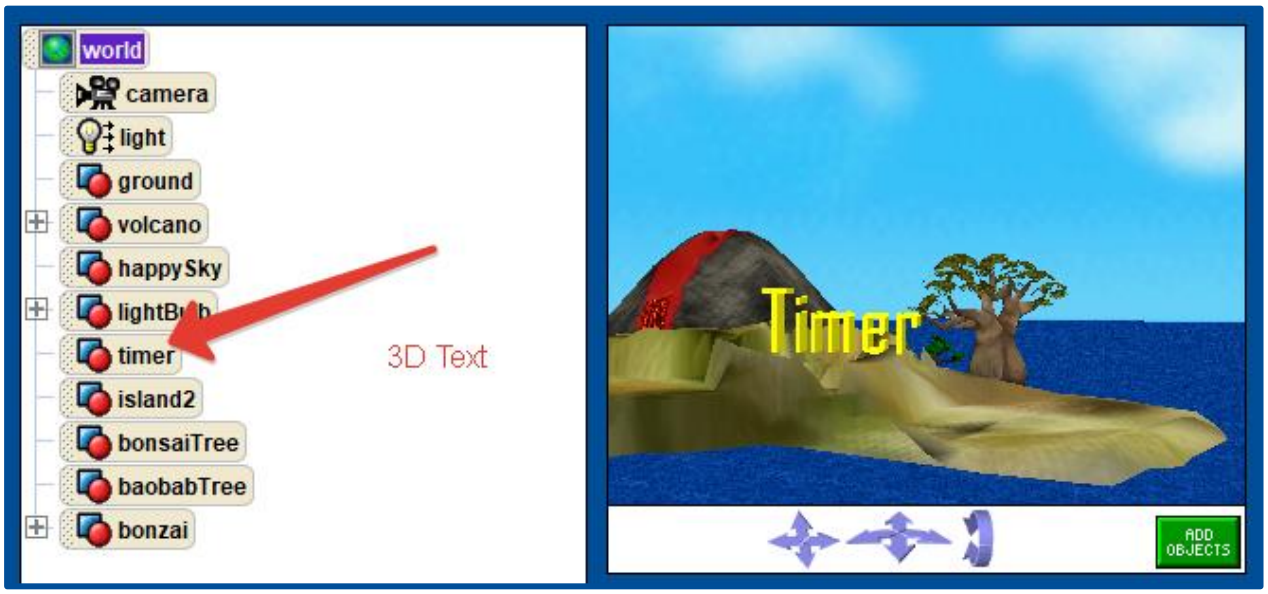


Рис. 199. Расположение объектов для сцены

2. Для объекта **timer** создайте новый метод, который называется **init**. В качестве параметра он будет принимать значение числа (**number**), от которого будет происходить счёт. Как уже было описано ранее, выберите в

дереве объектов объект **timer,** перейдите на вкладку **methods,** нажмите кнопку **create new method** и назовите метод **init**.

Перейдите на вкладку кода метода **timer.init**. Нажмите на кнопку **create new parameter,** в появившемся окне «создание нового параметра» задайте имя **number**, тип **number,** значение по умолчанию **0**.

Для объекта **timer** перейдите на вкладку **properties** (свойства) и нажмите на кнопку **create new variable** (создание новой переменной). Получается, что таким образом мы создаем новое свойство для объекта. Назовите это свойство **time,** присвойте по умолчанию значение **1**.

Перетащите свойства **time** в окно кода метода **timer.init**. Произойдет вызов метода **set value** (установить значение) **number**, как показано на рисунке 200.

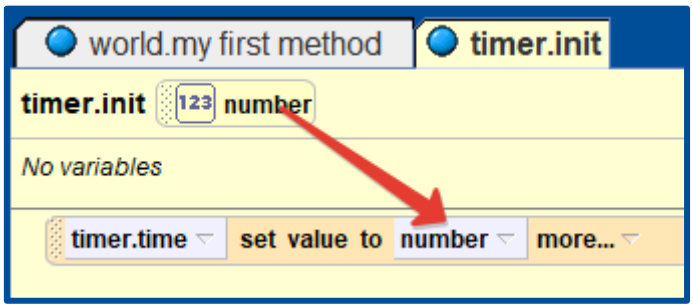


Рис. 200. Код timer.init

3. Для объекта **timer** создайте ещё один метод counDown. Данный метод должен работать по следующему алгоритму:

- **timer** устанавливает значение текст timer.time;
- пока timer.time > 0, делай:
  - уменьшай timer.time на 1
  - делай задержку 1 секунду
  - устанавливай **timer** устанавливает значение текст **timer.time** (рис. 201).

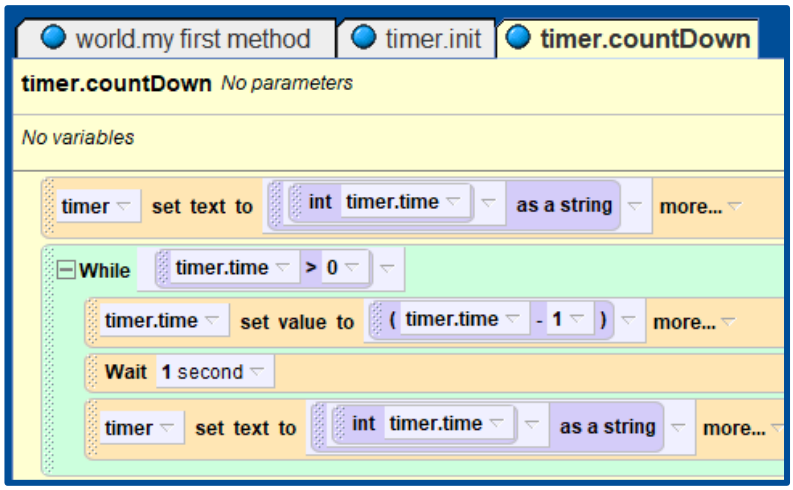


Рис. 201. Код метода timer.countDown

Для создания данного кода вам понадобятся некоторые функции. В коде они отмечены фиолетовыми участками. Чтобы их найти, выберите в дереве объектов world и перейдите на вкладку functions. Вам понадобятся what as a string (число перевести в строку), int a (число привести к целочисленному типу), a – b (разность двух чисел), a > b (сравнение чисел).

4. Создайте для вулкана метод **boom**, при котором происходит взрыв, камни появляются (свойство **isShowing** становится **true**) и лава течет.

5. В основной программе в my first method у вас будут последовательно вызваны 3 метода:

- timer.init (10);
- timer.countDown;
- volcano.boom.

6. Проверьте работу вашего мира.

7. Измените программу, чтобы таймер отсчитывался от 60, а задержка была 0,01 секунды.

Задание 2. Создайте проект, в котором **3D Text** будет показывать, сколько раз пользователь кликнул в мире и по какому объекту. Используйте для этого **Events** (событие), клик мышкой (**when the mouse is clicked on something**). Подобное событие было описано в лабораторной 2 (задание 2).

Задание 3. Сколько будет?

Создадим проект, проверяющий, правильно ли выполнено сложение двух чисел (числа выводятся случайным образом). Предлагать примеры на сложение программа должна до тех пор, пока с клавиатуры не введется число, которое означает выход.

В этом проекте мы будем оперировать случайными числами и закрепим работу с циклом **While**.

1. Создайте новый мир, поместите в него персонажа, например котенка (**cat**). Можно придумать интересную локацию.

2. Создайте новый метод **world.slozhenie** для создания примеров на сложение случайных чисел. Перетащите **world.slozhenie** в окно кода **my first method**.

3. Откройте на редактирование **world.slozhenie**. Для создания кода вам понадобится несколько функций: **random number**, **what as a string**, **a joined with b**. Эти функции можно найти, если в дереве объектов выбрать объект **world** и перейти на вкладку **functions** (рис. 202).

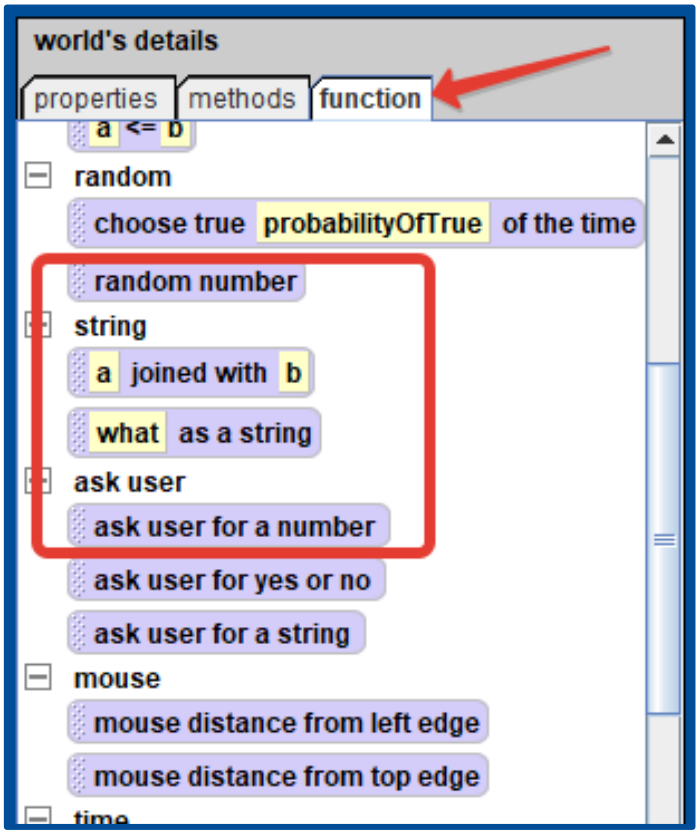


Рис. 202. Список необходимых функций

Также для генерации случайного числа нужно воспользоваться дополнительной опцией **more** для функции **random number**. На рис. 203 показано, как нужно действовать в этом случае.

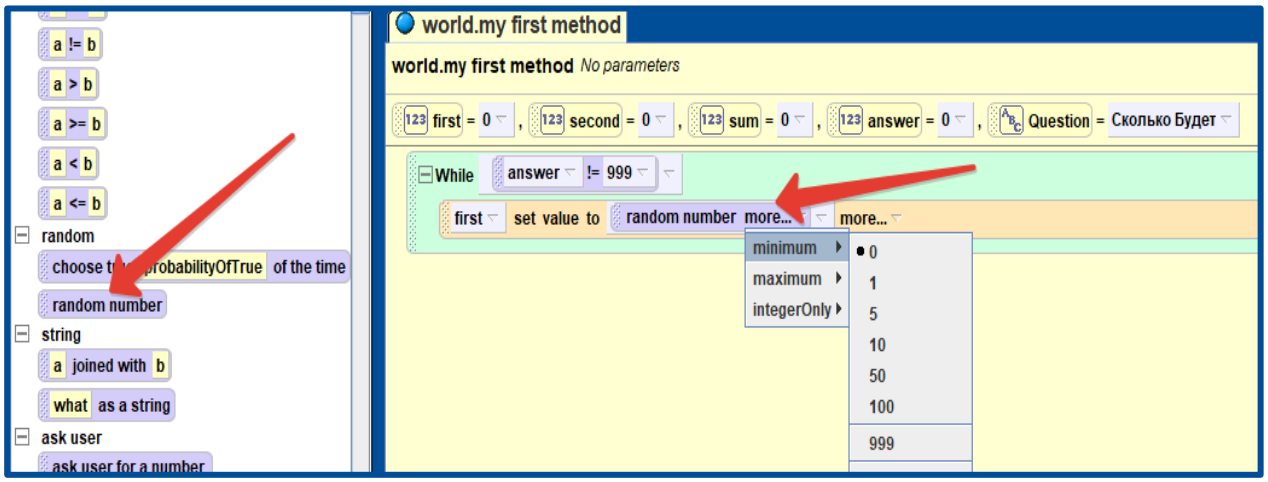


Рис. 203. Выбор начального и конечного значения для функции random number

Полный код метода показан на рис. 204.

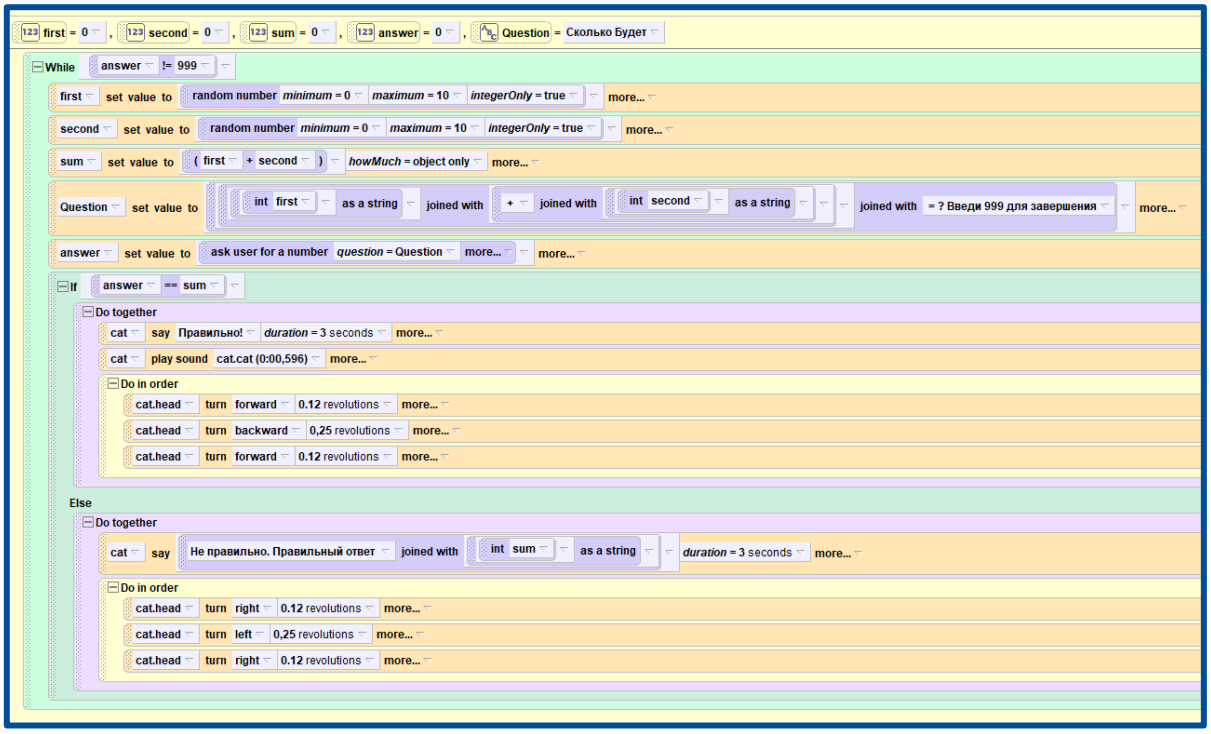


Рис. 204. Работа алгоритма сложения

Задание 4. Дополните алгоритм другими методами: умножением, вычитанием, делением на целое число. Реализуйте выбор операции (например, после ввода 999 программа не завершается, а происходит предложение выбрать операцию).

2.8. Лабораторная работа 8. Разработка алгоритмов движения персонажей в Alice

Чтобы наши персонажи «ожили», необходимо продумать им реалистичные движения, тогда мир виртуальной реальности будет в чем-то похож на настоящий. В этой лабораторной работе разработаем несколько алгоритмов, которые связаны с имитацией движения персонажей. Как вы уже убедились, все персонажи в Alice делятся на виды: двуногие, 4-х ногие, крылатые и другие.

Летающие персонажи, содержат подобъекты – крылья, которые при движении перемещаются вверх-вниз. При этом сам объект должен двигаться вперёд.

Персонажи двуногие двигают ногами и руками. Руки двигаются в такт перемещения персонажа с помощью ног.

Четырёхногие перебирают лапами при движении поочередно.

Разработка алгоритмов движения – важная составляющая при программировании роботов и беспилотных летательных аппаратов.

Задание 1. Передвижение летающего персонажа

Разработайте алгоритм полета ястреба (**hawk**). Ястреб в полете движется плавно, наблюдая за добычей. Крыльями машет вверх-вниз с небольшой амплитудой, одновременно продвигаясь вперед. Ястреб может кружить, то есть поворачиваться вокруг какого-то места.

Создайте методы поворота ястреба направо, налево и движения вперед на 1 шаг. С помощью событий Events сделайте управление движением ястреба (нпаример, стрелка вправо – поворот ястреба направо, стрелка влево – поворот налево, пробел – движение прямо). Примерный вид мира показан на рис. 205.

Рис. 205. Полет ястреба

Задание 2. Передвижение водоплавающего персонажа

Создайте подводный мир, в котором персонажами являются рыбки. Разработайте движение рыбки вокруг коралла. Рыбка движется плавно, но при этом резко и круто (на большой угол) может развернуться. Во время движения у рыбки плавно двигаются плавники. При повороте поворачивается хвост. Используйте цикл **Loop** для многократного повторения движения рыбки.

Чтобы у вас получился настоящий подводный мир, найдите все объекты в папке **Ocean**. В этой же папке есть локация **Ocean Floor**, которую можно перетащить в любой базовый мир (рис. 206).

Рис. 206. Подводный мир

Задание 3. Передвижение двуногого персонажа

Рассмотрите на рис. 207 схему передвижения двуногого персонажа. Обратите внимание, что при движении двигаются и ноги, и руки.

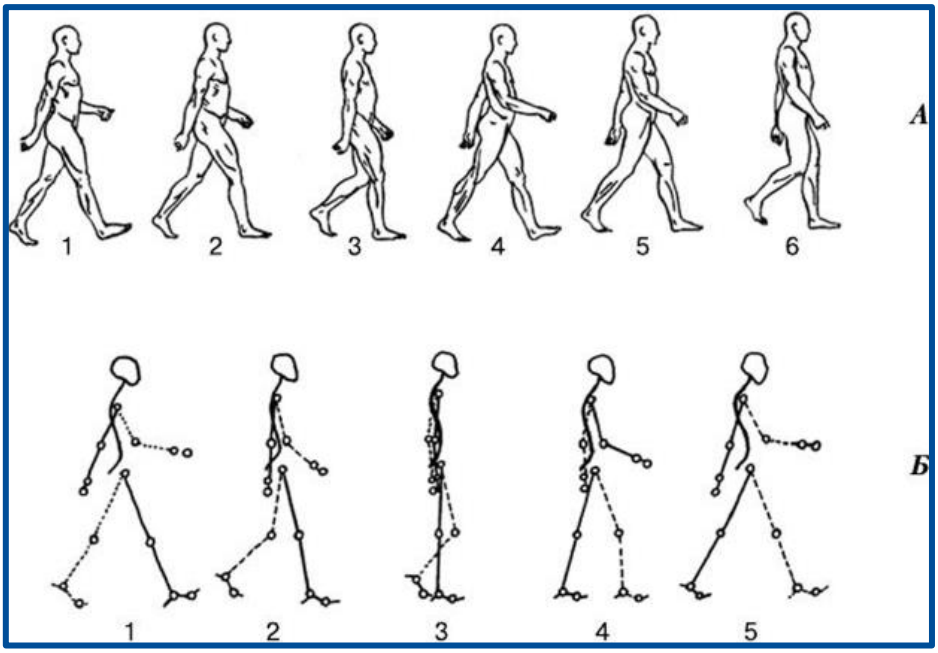


Рис. 207. Схема передвижения двуногого персонажа

Выберите любого двуногого персонажа: человека, мумию, инопланетянина и пр. Разработайте для него одну итерацию движения (шаг левой, шаг правой). Сделайте цикл из 10–15 шагов.

Задание 4. Передвижение четвероногого персонажа.

Разработайте алгоритм бега четвероногого персонажа, например: собаки, лошади, льва (рис. 208).

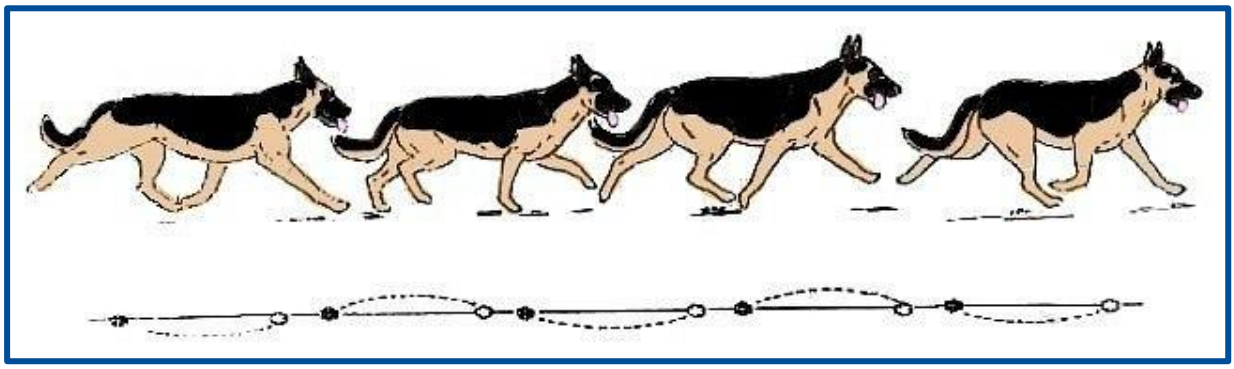


Рис. 208. Схема бега собаки

2.9. Лабораторная работа 9. Реализация алгоритмов преследования/уклонения персонажей в Alice

Задание 1. Случайное блуждание лунохода

Разработайте проект, в котором луноход совершает случайное блуждание. При этом создайте условие, при котором с изменением направления движения может случайным образом измениться и скорость движения.

Задание 2. Алгоритм преследования

Разработайте проект, в котором астронавт догоняет сбежавший ровер (луноход). У лунохода сломалась программа, и он стал убегать от астронавта. Причем, понятно, что астронавт рано или поздно догонит луноход.

Но вот незадача – луноход постоянно меняет направление: то налево повернет, то направо. У лунохода есть методы «Поворот направо», «Поворот налево». Чтобы обозревать все игровое поле, поднимите камеру над поверхностью **ground**, чтобы была хорошая видимость.

На рис. 209 представлен код программы, в котором вы самостоятельно должны написать методы для лунохода, – это **turn_left** и **turn_right** (рис. 209).

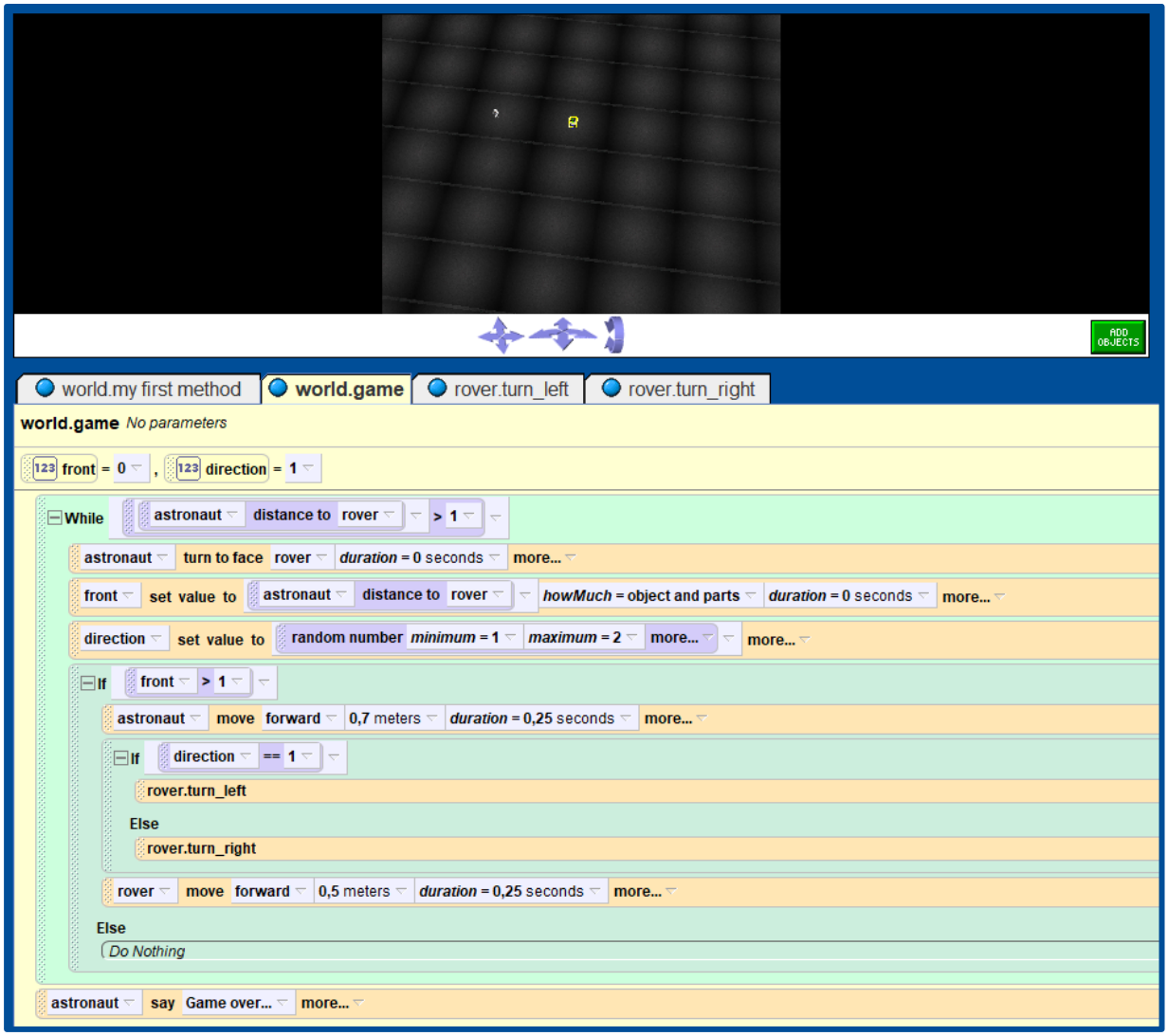


Рис. 209. Алгоритм преследования

Задание 3. Алгоритм уклонения

Луноход совсем сломался. В какой-то момент он перестал убегать от своего астронавта и начал его преследовать. Модифицируйте программу,

чтобы астронавт уклонялся от лунохода. Для управления астронавтом используйте такие клавиши: стрелка влево – поворот астронавта налево, стрелка вправо – поворот астронавта направо.

2.10. Индивидуальные задания по программированию в Alice

Проект должен иллюстрировать законченный сюжет (нельзя обрывать сюжет на середине). Надо использовать как можно больше возможностей Alice (количество объектов, собственных методов, управление камерой с помощью мыши, клавиш, работа с освещением, различными алгоритмическими конструкциями).

Темы проектов

1. **Дрессированный персонаж.** Придумайте для четвероногого персонажа (например, собаки, кошки, лошади) 4 движения (например, сидеть, лежать, сахарок и др.). Запрограммируйте в виде методов эти движения. Управление организуйте с помощью клавиш на клавиатуре.

2. **Двоичная система счисления (СС).** Разработайте алгоритм, с помощью которого с детьми можно отрабатывать перевод из десятичной СС в двоичную. Используйте для реализации случайные числа и проверку правильности введенного результата от пользователя.

3. **Танец.** Разработайте алгоритм, в котором балерины (куклы, животные или любые другие персонажи) исполняют танец. Объекты можно помещать в списки. Используйте комбинацию алгоритмических конструкций **for all in order** (для всех по порядку), **for all together** (для всех вместе).

4. **Сложение дробей.** С помощью персонажей Alice разработайте увлекательную игру, в которой объясняется правило сложения обыкновенных дробей. Проверьте знания пользователя, задавая вопросы на сложения случайных обыкновенных дробей и считая правильные и неправильные ответы. Для иллюстрации примеров и набранных очков за правильные и неправильные ответы можно использовать в качестве объекта **3D-текст**.

5. **Собери игрушки.** Разработайте игровое приложение, в котором персонажу Alice нужно собрать разбросанные игрушки. Игрушки появляются и исчезают через несколько секунд. Алисой можно управлять с помощью клавиш управления курсором. С помощью объекта **3D-текст** выводится количество собранных игрушек.

6. **Перекресток.** Разработайте сюжет, в котором дети могли бы научиться правильно переходить через дорогу по светофору. Для светофора используйте лампочки. Им можно задавать любой цвет, делать включенными, отключенными. Используйте в качестве объектов транспортные средства, дома, другие объекты города для создания максимально приближенной к реальности сцены.

7. **Тренажер английского языка.** Придумайте локацию (классный кабинет, городская среда, комната, лес и пр.), в которой расположите различные предметы. Игра заключается в том, что персонаж просит вас найти предмет и сообщает название этого предмета на английском языке. Нужно кликнуть по данному предмету. Программа должна считать количество правильных ответов.

8. **Уклонись от молнии.** Разработайте игру, в которой в небе движется облако с молнией. Оно движется вслед за персонажем и периодически показывает молнию. Вам нужно уклонять от молнии персонажа. Управление персонажем происходит с помощью клавиш «влево» (←) и «вправо» (→).

Программа должна заканчиваться, когда количество молний, попавших в персонажа, становится равным 10.

9. **Робот-пылесос.** Робот пылесос может работать в пределах какой-либо площадки. Площадку можно обозначить стенами (плоскостями, кубиками, параллелепипедами из папки **Shapes**). Если встречает ограничение области площадки, то он должен отъехать немного назад, повернуться на небольшой угол и продолжить движение в другом направлении до тех пор, пока снова не встретит край площадки. Когда робот-пылесос натыкается на объект, то пылесосит этот объект. Количество почищенных объектов увеличивается на единицу, это фиксирует счетчик. Игра заканчивается, когда количество почищенных объектов становится равным 5.

10. **Музыкальное произведение.** Разработайте проект, в котором пользователю нужно повторить мелодию. Букве на клавиатуре соответствует нота. Сначала пользователю проигрывается мелодия, потом он должен ее повторить, нажимая нужные буквы. Если повторил правильно, то засчитывается очко, если нет, то начисляется штрафной балл. Используйте метод **Play sound**, который подходит для любого объекта. Аудиофайлы можно импортировать, можно записать свои.

11. **Бинарный поиск.** Разработайте проект, в котором персонаж пытается отгадать задуманное вами число методом половинного деления. По правилу игры вы можете загадать любое число от 1 до 100. Персонаж сам генерирует число (чаще всего это середина отрезка от 1 до 100, то есть число 50). Если загаданное число меньше 50, то дальше персонаж будет искать методом деления пополам отрезка от 1 до 50. Если же число больше 50, то искать его персонаж будет на отрезке от 50 до 100. Игра продолжается пока персонаж не найдет загаданное число. В игре нужно также подсчитывать, за какое число шагов искомое значение будет найдено.

12. **Переливающиеся лампочки.** Разработайте проект, в котором используется световое представление – иллюминация. В проекте разместите лампочки в форме квадратной матрицы 5x5. Группы лампочек должны загораться в виде различных фигур (паттернов): ромбов, крестов, диагоналей. Разработайте 4–5 паттернов.

ГЛАВА 3. ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ И ПРОВЕРОЧНЫЕ МАТЕРИАЛЫ

3.1. Вопросы к зачету

1. Объясните понятие виртуальной реальности.
2. Объясните понятие киберпространства.
3. Что понимается под системами телеуправления?
4. В чем отличие виртуальной реальности от других способов получения информации?
5. Объясните термин «ограниченность виртуальной реальности».
6. Виды систем виртуальной реальности.
7. Приведите примеры систем виртуальной реальности для обучения сложным профессиям.
8. Приведите примеры систем виртуальной реальности для медицины.
9. Приведите примеры систем дополненной реальности.
10. Каковы преимущества систем виртуальной реальности при обучении и исследований?
11. Каковы особенности виртуальной реальности?
12. В чем состоят проблемы реализации систем виртуальной реальности?
13. Особенности визуализации графики в системах виртуальной реальности.

14. Особенности реализации физических законов в системах виртуальной реальности.

15. Что такое стереоизображение? Какие существуют технологии стереосъемки?

16. Какие существуют библиотеки для создания 3D-графики?

17. Особенности реализации систем искусственного интеллекта в системах виртуальной реальности.

18. Что понимается под детерминированными алгоритмами?

19. Что такое шаблоны поведения и как их можно применить в VR-системах?

20. Как использовать деревья планирования для имитации целенаправленной деятельности персонажа?

21. Эволюция аудио систем в VR-системах.

22. Особенности манипуляторов в VR-системах.

23. Проблема создания цифрового обоняния.

24. Что понимается под коллизией объектов?

25. В чем суть программной реализации алгоритма преследования?

26. В чем суть программной реализации алгоритма уклонения?

27. Особенности моделирования физики в BP.

28. Программная реализация изменения скорости движения персонажа.

29. Основные окна среды Alice

30. Каким образом добавляются объекты в среде Alice?

31. Особенности написания кода программы в среде Alice.

32. Назначение оператора Do Together.

33. В чем состоит назначение метода Do In Order?

34. Особенности программного управления персонажем в среде Alice.

35. Каким образом можно организовать диалог персонажей в среде Alice?

36. Ветвления и циклы в среде Alice.

37. Постановка и изменение координат камеры в среде Alice.

38. Назначение и особенности управления списками в среде Alice.

39. Обработка массивов в среде Alice.

40. Управление персонажами с помощью клавиатуры и мыши в среде Alice.

3.2. Примерные тестовые задания по виртуальной реальности

Вопрос 1. Термином «виртуальная реальность» обозначают:

1) созданный биолого-техническими средствами мир, передаваемый человеку (посетителю этого мира) через его ощущения: зрение, слух, обоняние, осязание и другие;

2) киберпространство, сеть Internet;

3) созданный техническими средствами мир, передаваемый человеку (посетителю этого мира) через его ощущения: зрение, слух, обоняние и другие;

4) совокупность информационных потоков, влияющих на человеческое сознание.

Вопрос 2. Киберпространство – это:

1) сеть Internet;

2) глобальная многопользовательская среда;

3) виртуальная реальность;

4) система с искусственным интеллектом.

Вопрос 3. Установите соответствие:

1) Вильям Гиббсон;
2) Рей Брэдбери;
3) Фридрих фон Лейбниц;
4) Айвен Сазерленд.

a) описал глобальную многопользовательскую среду в романе «Нейроман»;

b) описал вселенную монад – прототип глобальной сети;

c) поднял вопрос о влиянии информационных потоков на детскую психику в рассказе «Вельд»;

d) разработал прототип системы виртуальной реальности.

Вопрос 4. Установите соответствие:

1) Маршалл Маклюэн;
2) Иммануил Кант;
3) Карл Поппер;
4) Станислав Лем.

a) рассматривал мир как совокупность трех миров: объективный мир с физическими свойствами; субъективный мир сознания; всем доступный мир, являющийся отображением физического мира;

b) в работе «Понимание медиа» поднял вопрос о влиянии средств массовой информации;

c) в произведении «Повторение» подчеркнул идею ограниченности любого искусственного мира и невозможности искусственного создания идеального мира;

d) рассматривал реальность во взаимосвязи со временем; отмечал, что невозможно представить реальность без пространства.

Вопрос 5. Установите соответствие:

1) системы расширенной реальности;	a) специальное оборудование позволяет ощущать наряду с реальным и виртуальный мир (используется для развлечений, в трансляции спортивных соревнований);
2) системы телеприсутствия;	b) изменение картинки перед глазами с учетом движения головы;
3) визуально-согласованный дисплей.	c) позволяют человеку погрузиться в другую удаленную среду с использованием дополнительной аппаратуры.

Вопрос 6. Отличительной чертой всех систем виртуальной реальности является:

1) эффект погружения;
2) 3D-графика;
3) наличие дополнительных устройств ввода-вывода информации;
4) ответ на воздействия пользователя (интерактивность).

Вопрос 7. Технология создания виртуальных туров – это:

1) обладающий интерактивностью просмотр панорамных фотографий высокого разрешения;

2) обладающий интерактивностью просмотр видеозаписей высокого качества;

3) полностью аналогичная реальному месту созданная на компьютере 3D-сцена, обладающая возможностью интерактивного изменения точек просмотра;

4) набор интерактивных стереоизображений.

Вопрос 8. На чем основан эффект стереоизображения?

1) на особенности бинокулярного человеческого зрения;

2) на технологии трехмерного моделирования;

3) на особенности псевдо фотографий 3D;

4) на технологии записи виртуальных туров.

Вопрос 9. Факторы, дающие информацию об объёме в обычной фотографии:

1) аккомодация хрусталика;

2) тени и блики;

3) движение наблюдателя и движение предмета наблюдения;

4) бинокулярность зрения – наличие не одного, а двух глаз. Мозг сопоставляет изображения, которые видит правый глаз и левый.

Вопрос 10. Факторы, дающие информацию об объёме в кино:

1) воздушная перспектива;

2) тени и блики;

3) движение наблюдателя и движение предмета наблюдения;

4) геометрическая перспектива и знание реальных размеров наблюдаемых объектов.

Вопрос 11. Среди перечисленных пунктов укажите те, которые НЕ относятся к возможностям просмотра стереоизображений:

1) анаглиф;

2) скрещенный взгляд;

3) параллельный взгляд;

4) перекрестный взгляд;

5) поляризация;

6) затворные очки;

7) виртуальный шлем;

8) 3D мониторы;

9) 3D экраны.

Вопрос 12. Анаглиф – это:

1) просмотр стереоизображений с использованием двухцветных линз;

2) просмотр стереоизображений с использованием поляризационных линз;

3) просмотр стереоизображений с использованием миниэкранов для каждого глаза.

Вопрос 13. 3D-графика представляет собой:

1) растровую графику;

2) растровую и векторную графику;

3) векторную графику;

4) 3-х мерную графику.

Вопрос 14. В основе 3D модели лежит:

1) «проволочный каркас» из масштабируемых линий или многоугольников;

2) тело вращения;

3) полигональная сетка;

4) множество точек.

Вопрос 15. Установите правильную последовательность этапов создания 3D-графики:

1) освещение;

2) моделирование;

3) анимация;

4) съемка;

5) визуализация;

6) текстурирование.

Вопрос 16. Укажите формулу для расчета альфа прозрачности изображения:

1) R = B*(1 – T) + F*T, где B – исходный цвет пиксела, F – цвет накладываемого пиксела, T – прозрачность накладываемого пиксела [0..1];

2) R = B*(1 – F) + F*T, где B – исходный цвет пиксела, F – цвет накладываемого пиксела, T – прозрачность накладываемого пиксела [0..1];

3) R = (1 – F) + F*T*B, где B – исходный цвет пиксела, F – цвет накладываемого пиксела, T – прозрачность накладываемого пиксела [0..1].

Вопрос 17. Под z-буферизацией понимают:

1) алгоритм заливки трехмерного объекта;

2) способ учёта удалённости элемента изображения, один из вариантов решения «проблемы видимости»;

3) способ рендеринга сцены;

4) алгоритм расчета прозрачной области.

Вопрос 18. Уставите соответствие между моделью расчета затененности объекта и характеристикой производительности:

1) плоская модель затенения;	a) самый быстрый метод, конечный объект не производит впечатление гладкой поверхности;
2) метод Гуро;	b) средний по скорости метод, конечный объект содержит помехи в области блика;
3) метод Фонга.	c) самый медленный по скорости метод, конечный объект производит впечатление гладкой поверхности.

Вопрос 19. Спрайт в компьютерной графике – это:

1) растровое изображение, свободно перемещающееся по экрану, наблюдение под несоответствующим углом приводит к разрушению иллюзии;

2) векторное 3D-изображение, свободно перемещающееся по экрану;

3) специальная программа, являющаяся частью графического конвейера, обеспечивающая конечную доработку различных эффектов освещения.

Вопрос 20. Установите соответствие

1) OpenGL	a) программный интерфейс приложения, поддерживающий возможность работы не только с графикой, но и звуком, устройствами ввода
2) DirectX	b) графическая библиотека, позволяющая скрыть аппаратные возможности различных платформ для более быстрой разработки интерактивных графических приложений

Вопрос 21. Под искусственным интеллектом понимается:

1) наука и технология создания интеллектуальных машин, особенно интеллектуальных компьютерных программ;

2) научное направление, в рамках которого ставятся и решаются задачи проектирования интеллектуальных интерфейсов приложений;

3) наука и технология создания искусственного разума в виде мозгового интерфейса, робототехники и пр.

Вопрос 22. Установите соответствие:

1) простые детерминированные алгоритмы;	a) случайный выбор определенного алгоритма выполнения подзадачи;
2) шаблоны и сценарии;	
3) поведение и состояние.	

b) набор случайных чисел определяет возможную стратегию персонажа;

c) случайное движение, преследование, уклонение.

Вопрос 23. Выбор шаблона движения в VR-системах может осуществляться с помощью:

1) оператора варианта;
2) цикла;
3) рекурсивного оператора;
4) динамических структур.

Вопрос 24. Алгоритмическое проектирование индивидуальности персонажей осуществляется на основе

1) вероятностей выбора того или иного действия персонажем
2) такое неосуществимо на программном уровне
3) нейронных сетей
4) дерева планирования

Вопрос 25. Продукция представляет собой:

1) логическое утверждение с рядом посылок и следствием;
2) арифметическую операцию сложение;
3) высокоуровневый план;
4) организацию шаблона движения.

Вопрос 26. Среди перечисленных методов нахождения пути выберите наименее эффективный:

1) обход по контуру;
2) поиск в ширину;
3) двунаправленный поиск;
4) алгоритм Дейкстры.

Вопрос 27. Среди перечисленных методов нахождения пути выберите наиболее эффективный:

1) алгоритм A* (A стар);
2) алгоритм Дейкстры;
3) поиск в глубину;
4) двунаправленный поиск.

Вопрос 28. Какую функцию выполняет нейрон в нейросетях?

1) обработать первый входной сигнал и послать выходные сигналы;
2) обработать входной сигнал;
3) обработать входной сигнал и послать или не послать выходной сигнал;
4) обработать входной сигнал и послать выходной сигнал.

Вопрос 29. Среди перечисленных областей укажите область применения нейронных сетей:

1) программирование;
2) теория машинного обучения;
3) логическая основа компьютера;
4) оптимизация процессов.

Вопрос 30. Метод вычислений, основанный на биологических моделях эволюции – это:

1) генетические алгоритмы;
2) нечеткая логика;
3) генетическая математика;
4) теория нейронных сетей.

Вопрос 31. Установите правильную последовательность:

1) выбор решения (обычно лучшие особи имеют большую вероятность быть выбранными), к которому применяются «генетические операторы»;
2) формулировка задачи, решение которой может быть закодировано в виде вектора («генотипа») генов, где каждый ген может быть битом, числом или неким другим объектом;

3) создание множества генотипов начальной популяции;

4) получение новых решений.

Вопрос 32. Критерием остановки генетического алгоритма НЕ является:

1) отсутствие возможности генетического скрещивания;

2) нахождение глобального, либо субоптимального решения;

3) исчерпание числа поколений, отпущенных на эволюцию;

4) исчерпание времени, отпущенного на эволюцию.

Вопрос 33. Метод анализа данных, элементы которых могут входить во множества частично:

1) нечеткая логика;

2) генетические алгоритмы;

3) теория нейронных сетей.

Вопрос 34. Среди перечисленных устройств укажите устройства ввода, имеющие 6 степеней свободы (3 пространственные координаты, 3 угла для описания ориентации тела в пространстве):

1) штурвал;

2) сенсорная перчатка;

3) джойстик;

4) мышь;

5) руль;

6) геймпад;

7) троттл.

Вопрос 35. Под степенью свободы устройства ввода понимается:

1) минимальное количество независимых переменных, необходимых для полного описания движения механической системы;

2) максимальное количество независимых переменных, необходимых для движения механической системы;

3) координаты манипулируемого тела в пространстве;

4) размерность пространства, в котором происходит манипулирование.

Вопрос 36. Среди перечисленных устройств укажите устройства ввода, имеющие 3 степени свободы:

1) сенсорная перчатка;
2) мышь;
3) костюм виртуальной реальности;
4) экзоскелет.

Вопрос 37. Укажите верный способ добавления объектов в мир Alice:

1) нажатие на кнопку Add Objects в окне отображения сцены;
2) в главном меню команды File – Add All Objects;
3) в контекстном меню команды Add Objects дерева объектов;
4) все перечисленные способы.

Вопрос 38. В чем состоит назначение метода Do Together в среде Alice?

1) выполнение действий одновременно несколькими объектами (параллельно);
2) выполнение действий совместно с ядром программы;
3) выполнение последовательных действий, одного за другим;
4) такого метода нет в среде Alice.

Вопрос 39. В чем состоит назначение метода Do In Order в среде Alice?

1) выполнение действий одновременно несколькими объектами (параллельно);
2) выполнение последовательных действий, одного за другим;
3) выполнение действий совместно с ядром программы;
4) такого метода нет в среде Alice.

Вопрос 40. В чем состоит назначение метода Loop в среде Alice?

1) добавление объекта в виртуальный мир;
2) это организация циклических действий по некоторому условию;
3) это организация циклических действий определенное количество раз;
4) выполнение совместных действий несколькими объектами.

Вопрос 41. Среди представленных методов управления камерой выберите тот, который отсутствует в среде Alice:

1) движение объекта мышью;
2) когда начинается мир;
3) когда объект появляется в кадре;
4) когда нажата клавиша;
5) когда нажимаешь мышью по объекту;
6) пока что-то является истинным;
7) когда переменная меняется;
8) движение объекта клавишами (стрелками);
9) управление камерой с помощью мыши;
10) ориентирование камеры на мышь.

Вопрос 42. Укажите верный синтаксис цикла, выполняющего действия для каждого объекта из серии:

1) While index < count;
2) For i := <start_position> to <end_position> do;
3) For each in;
4) For item in index do;

Вопрос 43. В цикле While в среде Alice действие может выполняться, если:

1) условие истинно или ложно (можно настраивать);
2) условие ложно;
3) условие истинно.

Вопрос 44. Цикл с параметром в среде Alice начинается с ключевого слова:

1) Loop;
2) Foreach;
3) Count;
4) For each;
5) For.

3.3. Темы докладов по виртуальной реальности

Ориентировочное время выступление – 10 минут. Необходимо подготовить интерактивную презентацию с теоретическими и графическими материалами, рекомендуется использовать видеоинформацию, подробные технические данные. К рекомендуемым источникам можно и нужно добавлять свои. Источники информации обязательно указать в презентации. Приветствуется использование иностранных источников.

По окончании выступления планируется обсуждение доклада. Выступающий должен быть компетентен в своей теме.

Вариант 1. Системы виртуальной реальности в медицине.

Источники:

- http://ve-group.ru/otrasli/meditsina/
- http://geektimes.ru/post/246228/
- https://www.google.ru/url?sa=t&rct=j&q=&esrc=s&source=web&cd=20&ved=0CFcQFjAJOApqFQoTCL7Q59j0xcgCFUESLAodsXQPFA&url=http%3A%2F%2Frosomed.ru%2Fkniga%2Fistoria_simulationnogo_obucheniya.pdf&usg=AFQjCNFgs-8my1_eWd7QBgBkss47Rdcqhg&sig2=63EdtIlk9nRrCxHVHKYimg

Вариант 2. Бесконтактное управление.

Источники:

- http://infoglaz.ru/?p=33446
- http://innogest.ru/m?na=12060

Вариант 3. Системы виртуальной реальности в подготовке пилотов летательных аппаратов.

Источники:

- http://vpk-news.ru/articles/23149
- http://ve-group.ru/3dvr-resheniya/trenazheryi/
- http://www.transas.ru/Simulation/Aviation

Вариант 4. Системы виртуальной реальности в геологии.

Источники:

- http://www.professionalgroup.ru
- http://www.professionalgroup.ru/produkcziya/3d-animacziya-i-grafika.html
- http://ve-group.ru/3dvr-resheniya/neft-i-gaz/

Вариант 5. Особенности графической библиотеки DirectX (графика, звук, обработка видео данных).

Источники:

- Адамс Д. DirectX: продвинутая анимация. Комплект. – «КУДИЦ-ПРЕСС», 2004. – С. 480. – ISBN 5-9579-0025-7
- https://habr.com/ru/post/489282/

Вариант 6. Искусственные нейронные сети в поисковых алгоритмах (Яндекс, Google). Рассмотреть применение нейронных сетей при обработке поисковых запросов, например: алгоритм «Королев» в системе «Яндекс», обработка устной речи в системе Google.

Источники:

- https://habr.com/ru/company/yandex/blog/314222/
- https://yandex.ru/blog/company/algoritm-palekh-kak-neyronnye-seti-pomogayut-poisku-yandeksa
- https://proglib.io/p/10-instrumentov-iskusstvennogo-intellekta-google-dostupnyh-kazhdomu-2020-03-03

Вариант 7. Применение нейронных сетей при распознавании образов. Рассмотреть возможности обучения и применения алгоритмов, основанных на нейронных сетях для распознавания дорожных знаков (автопромышленность), распознавания автомобильных номеров (контроль за безопасным движением на дорогах), распознавания лиц, улыбок (фототехника, изображения), и др.

Источники:

- https://habr.com/ru/post/74326/
- https://center2m.ru/ai-recognition
- https://habr.com/ru/company/recognitor/blog/225913/

Вариант 8. Компьютерная графика в киноиндустрии.

Источники:

- https://habr.com/ru/post/409317/
- https://3dnews.ru/167126

Вариант 9. Технология Motion Capture в киноиндустрии.

Источники:

- https://habr.com/ru/company/plarium/blog/366803/
- https://habr.com/ru/news/t/510678/
- https://dtf.ru/cinema/109583-kak-rabotaet-motion-capture-evolyuciya-i-zakulise

Вариант 10. Роботизированная техника на производстве (промышленные роботы).

Источники:

- https://habr.com/ru/company/top3dshop/blog/403323/
- https://3dtool.ru/stati/promyshlennye-roboty-dlya-proizvodstva/

Вариант 11. Роботизированная техника в быту (бытовые роботы).

Источники:

- https://habr.com/ru/company/vk/blog/401555/
- https://habr.com/ru/company/selectel/blog/522134/
- https://habr.com/ru/company/selectel/blog/531962/

Вариант 12. Физические воздействия в VR-системах (физические движки Havok, PhysX).

Источники:

- https://www.havok.com
- https://pikabu.ru/story/chto_takoe_dvizhok_nvidia_physx_i_dlya_chego_on_nuzhen_3532717

Вариант 13. VR-системы при архитектурном проектировании.

Источники:

- https://vr-app.ru/blog/architecture/
- https://virtualnyeochki.ru/stati/virtualnaya-realnost-v-arxitekture

Вариант 14. VR-системы в телеуправлении.

Источники:

- https://habr.com/ru/post/438198/
- https://habr.com/ru/company/icover/blog/39147/
- https://habr.com/ru/post/563994/

ЗАКЛЮЧЕНИЕ

Виртуальная реальность, несмотря на прогнозируемые для нее перспективы, все же не стала системой для бытового потребления. Тем не менее человечество не останавливает попыток создания новых видов интерфейсов для управления виртуальной реальностью. Появляются новые технологии, которые обеспечивают для разработчиков низкий порог входа в индустрию. Такими средствами разработки в последнее время стали системы Unity 3D и Unreal Engine. Удешевляется VR-оборудование. Ускоряются вычислительные возможности компьютеров, что способствует созданию более реалистичной графики и компьютерной физики. Развиваются технологии искусственного интеллекта, что делает ВР сложной и интересной. Интерес к VR то растет (как это произошло в 2012–2015 г.), то постепенно снижается, уступая приоритет в финансировании других способов взаимодействия с информацией.

Научить взаимодействовать с VR и создавать VR-системы можно даже школьников. В образовании VR имеет большие перспективы. В частности, материалы для лабораторных работ из этого учебного пособия можно использовать и для обучения школьников программированию виртуальной реальности (например, в рамках дополнительного образования).

Технологии, которые изначально разрабатывались для VR, нашли свое применение в смежных отраслях: робототехнике, разработке искусственного интеллекта, математическом моделировании, автоматизации управления и др. Поэтому будущие поколения должны знать возможности VR-системы, уметь взаимодействовать с ней и извлекать из этого взаимодействия практическую пользу.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. DirectX 12 – от Леонардо да Винчи к современному искусству. – URL:https://habr.com/ru/post/489282/ (дата обращения: 24.08.2022).

2. Course Alice. – URL: https://www2.cs.duke.edu/csed/alice/aliceInSchools/ (дата обращения: 24.08.2022).

3. Kangaroo and turtle. – URL: https://www2.cs.duke.edu/csed/alice09/tutorials/gettingStartedTutorials/methodsTutorial/methodsTutorial.pdf (дата обращения: 24.08.2022).

4. Microsoft DirectX 11 – новый виток эволюции игровой графики. – URL: https://ru.gecid.com/video/microsoft_directx_11_novyeyi_vitok_yuvolyajii_igrovoyi_grafiki/?s=all (дата обращения: 24.08.2022).

5. Popular Physics Engines comparison: PhysX, Havok and ODE. – URL:http://physxinfo.com/articles/?page_id=154 (дата обращения: 24.08.2022).

6. Real-time monitoring of traffic parameters / K. Khazukov, V. Shepelev, T. Karpeta [et al.] // Journal of Big Data. – 2020. – Vol. 7. – No 1. – P. 84. – DOI 10.1186/s40537-020-00358-x. – EDN QXAOXE.

7. Tokamak. – URL: http://www.gamedev.ru/articles/?id=70125 (дата обращения: 24.08.2022).

8. Алгоритмические основы современной компьютерной графики. Лекция 9: Закрашивание. Рендеринг полигональных моделей. – URL:https://intuit.ru/studies/courses/70/70/lecture/2108?page=2 (дата обращения: 24.08.2022).

9. Бабенко, В.С. Виртуальная реальность: Толковый словарь терминов / В.С. Бабенко; ГУАП. – Санкт-Петербугр, 2006. – 87 с. – ISBN 5-8088-0165-6.

10. Бахтомин, Н.К. Теория научного знания Иммануила Канта: Опыт совр. прочтения «Критики чистого разума» / Н.К. Бахтомин. – Москва: Наука, 1986. – 205 с.

11. Брэдбери, Р. Вельд / Р.Бредбери // Сборник рассказов Рэя Брэдбери «Человек в картинках». – пер. Лев Жданов. – 1951.

12. Лейбниц, В. // Большая энциклопедия Кирилла и Мефодия. – Москва: Кирилл и Мефодий: Большая Российская энцикл., 2004. – 2 электрон. опт. диска (CD-ROM): ил.

13. Георгиевский, А. Алгоритмы 3D-графики / А. Георгиевский. – URL: http://abuse.edu.ioffe.ru/cluster/opengl_depth (дата обращения: 24.08.2022).

14. Гибсон, У. Нейромант / У. Гибсон // Одноим. авт. сб. - Москва: АСТ; Санкт-Петербург: Terra Fantastica, 1997.

15. Дацюк, С. Ноу-хау виртуальных технологий / С. Дацюк. – PC Club. – 1997. – №30.

16. Дацюк, С. Парадоксальные интенции свободы в Интернет / С. Дацюк. 1997.

17. Дашко, Ю.В. Основы разработки компьютерных игр в XNA Game Studio / Ю.В. Дашко, А.А. Заика. – URL: http://www.intuit.ru/department/se/xnagamestudio/ (дата обращения: 24.08.2022).

18. Ефимов, А.И. Методы применения нейронных сетей для оценки и повышения фотореалистичности виртуальной реальности / А.И. Ефимов // Инженерный вестник Дона. – 2019. – № 3(54). – С. 40. – EDN ZJETZD.

19. Зайцева, Н. Разработка физической модели разбиения твердого тела для игрового движка / Н. Зайцева. – URL: http://software.intel.com/ru-

ru/articles/Physical_modeling_of_destruction_for_game_engine/ (дата обращения: 24.08.2022).

20. Зинченко, Ю.П. Виртуальная реальность в экспериментальной психологии: к вопросу о методологии / Ю.П. Зинченко. – URL: http://vprosvet.ru/biblioteka/virtualnaya-realnost/ (дата обращения: 24.08.2022).

21. Как ведущие нефтяные и газовые компании используют виртуальную реальность. – URL:https://habr.com/ru/company/sibur_official/blog/568844/ (дата обращения: 24.08.2022).

22. Как работает альфа-композитинг. – URL:https://habr.com/ru/post/468067/ (дата обращения: 24.08.2022).

23. Козинцев, С. Способы просмотра / С. Козинцев. – URL: http://www.stereoart.ru/paper/p0014.html (дата обращения: 24.08.2022).

24. Королев, А.Л. Исследовательская деятельность будущих учителей информатики при изучении компьютерного моделирования / А.Л. Королев, Н.Б. Паршукова // Вестник Южно-Уральского государственного гуманитарно-педагогического университета. – 2020. – № 7(160). – С. 59–75. – DOI 10.25588/CSPU.2020.160.7.004. – EDN ZFOFZY.

25. Королев, А.Л. Компьютерное моделирование объектов, процессов и систем / А.Л. Королев, Н.Б. Паршукова. – Челябинск: ЮУрГГПУ, 2020. – 329 с. – ISBN 978-5-907409-15-6. – EDN XOTTPD.

26. Краткая история VR в 70-е – 80-е: военпром, интерактивные карты, опыт в играх и VR для NASA с бинауральным звуком. – URL: https://habr.com/ru/company/pult/blog/517296/ (дата обращения: 24.08.2022).

27. Краткая история VR: часть первая – ранние концепции и первые шаги от 1930-х до 1960-х. – URL:https://habr.com/ru/company/pult/blog/517050/ (дата обращения: 24.08.2022).

28. Ламот, А. Программирование игр для Windows: советы профессионала / А. Ламот. – 2004. – 873 с.

29. Лем, С. Собрание сочинений в двух томах / С. Лем. – Москва: Мир, 1992. – 304 с.

30. Лукьяненко, С. Лабиринт отражений / С. Лукьяненко. – Москва: АСТ; Санкт-Петербург: Terra fantastica, 1997.

31. Маклюэн, М. Понимание медиа: внешние расширения человека / М. Маклюэн / пер. с англ. В. Николаева; закл. ст. М. Вавилова. – Москва: Канон-пресс; Жуковский: Кучково поле, 2003. – 464 с.

32. Паршукова, Н.Б. Особенности реализации междисциплинарных связей в процессе изучения курса «Психолого-педагогические основы виртуальной реальности» / Н.Б. Паршукова // Информатизация образования: проблемы и перспективы: сб. науч. ст. – Челябинск: Цицеро, 2012. – С. 31–35. – EDN TVVKGH.

33. Паршукова, Н.Б. Создание и использование виртуальной лаборатории как средства формирования предметной компетенции по геометрии у учащихся основной школы: дис. канд. пед. наук / Н.Б. Паршукова. – Екатеринбург, 2009. – 205 с. – EDN NQONPN.

34. Паршукова, Н.Б. Формирование специальных компетенций при подготовке студентов специальности ИТВО в курсе «Психолого-педагогические основы виртуальной реальности» / Н.Б. Паршукова // Информатика и информационные технологии: сб. науч. ст. – Челябинск: Цицеро, 2013. – С. 148–151. – EDN TVHQWJ.

35. Первые шаги к виртуальной реальности. – URL: http://www.hwp.ru/Multimedia/I-glasses/indx.html?SHOWALL_1=1(дата обращения: 24.08.2022).

36. Петрова, Н. Виртуальная реальность как новый метод арт-терапии, или расставание с собой / Н. Петрова. – Москва, 1998.

37. Петрова, Н. Виртуальная сфера, или новые лица в пространстве виртуальной реальности / Н. Петрова // Мир ПК. – 1998. – № 02.

38. Петрова, Н. Перспективы виртуальной реальности / Н. Петрова // Мир ПК. – 1998, 1997.

39. Прохоров, А. Игровые манипуляторы / А. Прохоров // Компьютер пресс. – 2001. – № 6.

40. Семенов, А. Музыкальные форматы в играх / А. Семенов. – URL: http://shiru.untergrund.net/articles/musformats.htm (дата обращения: 24.08.2022).

41. Современные стандарты 3D-звука. – URL: http://art.thelib.ru/computers/raz/sovremennie_standarti_3dzvuka.html (дата обращения: 24.08.2022).

42. Статья «Как работает рендеринг 3D-игр: текстурирование и фильтрация текстур». – URL: https://habr.com/ru/post/499540/ (дата обращения: 24.08.2022).

43. Стоит ли брать VR-перчатки и как их выбрать: пошаговая инструкция + ТОП-6 лучших моделей. – URL: https://vr4you.ru/zhelezo-i-aksessuary/vr-perchatki (дата обращения: 24.08.2022).

44. Транзас электронные технологии, статьи. – URL:http://www.transas.ru/ (дата обращения: 24.08.2022).

45. Туториал по Unreal Engine. Часть 1: знакомство с движком. – URL: https://habr.com/ru/post/344394/ (дата обращения: 24.08.2022).

46. Усманова, Л.М. Разработка демонстрационной программы по виртуальному физическому эксперименту / Л.М. Усманова, Н.Б. Паршукова // Информатизация образования: проблемы и перспективы: мат-лы IV

Всероссийской науч.-практ. интернет-конф., посвященной памяти Д.Ш. Матроса, Челябинск, 12 апреля 2018 года / под общей ред. Г.Б. Поднебесовой. – Челябинск: ЮУрГГПУ, 2018. – С. 106–111. – EDN YQDZZZ.

47. Учить, лечить, воевать и сохранять: необычные применения технологии захвата движения. – URL:https://habr.com/ru/company/toshiba-rus/blog/516560/ (дата обращения: 24.08.2022).

48. Хилл, Ф. OpenGL программирование компьютерной графики. / Ф. Хилл. – Санкт-Петербург: Питер, 2002. – 1088 с.: ил. – ISBN: 5-318-00219-6.

49. Цифровые ароматы: запись, восстановление и передача запахов. – URL: https://habr.com/ru/company/vk/blog/407721/ (дата обращения: 24.08.2022).

50. Баяковский, Ю. Графическая библиотека OpenGL: метод. пособие / Ю. Баяковский, А. Игнатенко, А. Фролов. – URL:http://www.rsdn.ru/article/opengl/ogltut2.xml (дата обращения: 24.08.2022).

ПРИЛОЖЕНИЕ

ТЕЗАУРУС

Анаглиф (от греч. ἀνάγλυφος «рельефный») – метод сепарации изображений чёрно-белой стереопары при помощи цветового кодирования.

Разные части стереопары, предназначенные для левого и правого глаза, проецируются или печатаются на одну и ту же поверхность, но окрашиваются в разные цвета, дополнительные друг к другу. Чаще всего используются красный и бирюзовый цвета.

В красном изображении анаглифической стереопары содержится картина для левого глаза, а в бирюзовом (сине-зелёном) – для правого. При рассматривании такой стереопары через специальные анаглифические очки со светофильтрами тех же цветов наблюдается объёмное монохромное изображение, поскольку каждый глаз видит только свою часть стереопары.

Анимация (animation – оживление) – технология создания изображений движущихся объектов, основанная на принципах мультипликации.

При А. необходимый эффект достигается путем формирования и воспроизведения последовательности изображений, соответствующих различным фазам движущихся объектов. В системах мультимедиа и виртуальной реальности процедура А. обычно реализуется электронными средствами.

Аппаратные средства (англ. hardware) – совокупность физически существующих компонентов некоей электронной системы. А.с. иногда называют аппаратным обеспечением, а в разговорной речи – «железом».

В компьютерных системах и системах виртуальной реальности (СВР) в состав А.с. включают как системные блоки, реализующие основные вычислительные процедуры, так и периферийное оборудование. В частности, в СВР это головные дисплеи, управляющие перчатки и др. Возможности СВР по созданию виртуальных миров во многом определяются именно А.с.

Аппроксимация (англ. approximation) – процедура приближенного формализованного представления сложных геометрических многообразий через совокупность более простых. Примером может служить замена произвольных кривых ломаными линиями, состоящими из прямолинейных отрезков; замена произвольных криволинейных поверхностей системой сопряженных плоских граней, а тел сложной конфигурации многогранниками.

Процедура А. позволяет упрощать математическое описание объектов виртуального мира и, соответственно, создание системы виртуальной реальности.

Однако процедура А. вносит определенную погрешность в математические описания объектов. Эта погрешность тем меньше, чем больше использовано аппроксимирующих элементов. Например, при А. некоей замкнутой кривой многоугольником точность представления тем выше, чем больше сторон содержит этот многоугольник. Но при этом математические описания объектов и, соответственно, системы виртуальной реальности усложняются.

Обычно ищут компромисс между точностью представлений объектов и сложностью технической реализации при учете целей и задач, решаемых системой виртуальной реальности.

Бинауральный эффект (англ. binaural effect) – эффект восприятия звукового поля внешнего мира, обусловленный наличием у человека двух ушей. Б.э. основан на том, что из-за пространственного размещения ушей

человека время прохождения звуковых волн до каждого уха и фазовые соотношения неодинаковы. Причем различие левого и правого уха зависит от ориентации на источник звука.

Б.э. является важным фактором в определении человеком направления и расстояния до источника звука, в том числе и в СВР.

Бинокулярное зрение (англ. binocular vision) – зрение, обусловленное наличием у человека двух глаз, которые расположены в пространстве. Благодаря Б.з. человек воспринимает глубину пространства.

Основные факторы Б.з.: диспаратность и конвергенция. Диспаратность и угол конвергенции увеличиваются по мере приближения объектов к наблюдателю.

Наличие Б.з. создает у человека рельефную объемную картину видимого внешнего мира. Б.з. является важным фактором восприятия видимого пространства, особенно объектов, находящихся на близких расстояниях.

Боковое зрение (иногда называется периферическим) соответствует участкам поля зрения, видимым под углами, превышающими 40 – 60° от оси глаза по горизонтали. Отличается сравнительно низкой пространственной разрешающей способностью, но имеет

повышенную чувствительность к восприятию временных изменений (пульсаций) яркости.

Виртуальная реальность (ВР, англ. virtual reality, VR, искусственная реальность) – созданный техническими средствами мир, передаваемый человеку через его ощущения: зрение, слух, осязание и другие.

Виртуальная реальность имитирует как воздействие, так и реакции на воздействие. Для создания убедительного комплекса ощущений реальности компьютерный синтез свойств и реакций виртуальной реальности производится в реальном времени.

Визуализация (англ. visualization) – процесс преобразования невизуальной информации в визуальную форму. Обладает свойством виртуальности. Цель В. представить оператору человеко-машинной системы информацию в наглядной, нередко интегральной, обобщенной форме.

Благодаря В. информация становится доступной зрительному восприятию оператора, увеличивается объем и скорость переработки им информации.

Можно выделить две основные формы В.:

- преобразование объектов, представленных в невидимых глазом лучах (инфракрасных, ультразвуковых, рентгеновских и др.), в видимые изображения;
- преобразование некоторой совокупности (массива) цифровых данных в наглядную визуальную форму (в виде графиков, диаграмм, условных изображений и т.п.).

Виртуальный мир (англ. virtual world) – термин имеет два значения:

1. Мир, объекты и процессы которого обладают свойством виртуальности; соответственно, он может существовать вне сознания человека.

2. Мир виртуальной реальности, в который погружается и с объектами которого может взаимодействовать пользователь системы виртуальной реальности.

Виртуальный музей (англ. virtual museum) – музей, основу которого составляют виртуальные экспозиции и экспонаты. Известны две формы организации В.м.:

1. В.м., в которых экспозиции представлены на компакт-дисках или иных запоминающих устройствах с большим объемом памяти. Экспонаты в таком в.м. могут рассматриваться с помощью стандартных мониторов или проекционных устройств без иммерсии пользователя. К этому классу относятся в.м., экспозиции которых представлены на сайтах тех или иных музеев или организованы самостоятельно.

2. В.м., экспозиции которых находятся в мире виртуальной реальности. В этих случаях пользователь (посетитель) погружается в виртуальный мир, наблюдает объекты этого мира и может взаимодействовать с ними. При этом видимая пользователем виртуальная картина может отображать залы музея с экспонатами, по которым ходит посетитель, или же картину того мира, которая экспонируется в музее.

Генератор (англ. generator) – в данном случае подсистема вычислительного комплекса системы виртуальной реальности, формирующая сигналы, которые при воспроизведении образуют картину элементов и объектов виртуального мира. Можно различать Г. точек, линий, многоугольников, поверхностей, граней, многогранников, текстуры и изображений.

Грань (англ. verge) – элемент многогранника, представляющий собой участок плоскости, ограниченный многоугольником. Г. является основным аппроксимирующим элементом при математическом представлении поверхностей и тел сложной конфигурации.

3D-моделирование – это процесс разработки математического координатного представления любой поверхности объекта в трех измерениях с помощью специализированного программного обеспечения путем манипулирования ребрами, вершинами и полигонами в моделируемом 3D-пространстве.

Джойстик (англ. joystick – палочка радости) – устройство ввода информации в персональный компьютер. Представляет собой качающуюся в двух плоскостях вертикальную ручку.

Дополненная реальность (англ. augmented reality, AR – дополненная реальность) – воспринимаемая смешанная реальность, создаваемая с помощью компьютера с использованием «дополненных» элементов воспринимаемой реальности, когда реальные объекты монтируются в поле восприятия.

Захват движения (англ. motion capture) – метод анимации персонажей и объектов, при котором анимация создаётся не вручную, а путём оциф-

ровки (видеозаписи с помощью специальных датчиков) движений реального объекта (прежде всего, человека) и последующего переноса их на трёхмерную модель.

Динамическая VR-система (англ. motion-based VR systems) – системf, чаще всего транспортная, установленная на подвижном основании, управляемом в соответствии с визуальным содержанием сюжета.

В Д.с.в.р. иммерсия во многом определяется силовой обратной связью через подвижное основание. Участник (или группа участников) обычно играет роль пассажира некоего транспортного средства, управляемого реальным или виртуальным водителем. Применяется, в основном, в развлекательных целях как своеобразный аттракцион.

Динамический стереоэффект (англ. motion stereo effect) – эффект зрительного восприятия человеком объемности внешнего мира и скорости движения в нем, вызванного перемещением объектов в поле зрения человека. Д.с. возникает в случаях, когда человек движется в пространстве или же в его поле зрения находятся движущиеся объекты. Д.с. является важным фактором ориентирования в пространстве в случаях, когда пользователь находится в транспортном средстве, перемещающемся с большой скоростью.

Дискретизация (англ. discretization) – в общем случае замена непрерывно изменяющихся величин отдельными отсчетами через определенные интервалы, называемые шагом Д.

Процесс Д. сигналов и изображений является основополагающим в телевидении, вычислительной технике, в системах виртуальной реальности. Различают пространственную, временную и энергетическую Д.

При пространственной Д. пространство и находящиеся в нем объекты разбиваются на множество конечных минимальных элементов: пикселов (для двухмерного) и вокселов (для трехмерного пространства). В пределах пиксела и воксела энергетические показатели полагаются постоянными.

При временной Д. значения изменяющихся во времени процессов производят через определенные интервалы времени.

Энергетическая Д., обычно называемая квантованием, заключается в том, что значения энергетических параметров изображения или сигнала могут принимать лишь определенные, разрешенные значения. Введение Д. вносит определенные ограничения в передаваемую и воспроизводимую информацию (не воспроизводятся сколь угодно мелкие объекты, сколь угодно быстрые изменения, все возможные изменения энергии), а также вносит специфические помехи (альясинг, муар эффект, стробоскопический эффект). Ограничения и помехи тем меньше, чем меньше шаг Д. Но при этом заметно усложняются аппаратурные и программные затраты. Обычно шаг Д. определяют, исходя из целевого назначения системы, технических возможностей и учета порогов восприятия органов чувств человека или иного приемника информации.

Затенение (англ. shading) – нанесение на грани объектов синтезированного изображения полутеней в соответствии с их ориентацией, положением источников света и наблюдателя в пространстве. Благодаря этому объекты виртуального мира становятся рельефными и имеют более естественный вид. Существуют различные методы затенения.

Простейшим является так называемое плоское З. В этом случае поверхности каждой грани придается некая средняя яркость с учетом общей полутени. Но в этом случае на ребрах будут иметь место скачки яркости, подчеркивающие дискретность объектов, вносимую аппроксимацией.

Для устранения этого дефекта используются более сложные процедуры З., при которых яркость по поверхности граней меняется плавно, а скачки яркости на ребрах исключаются. Наиболее известны методы З. с использованием линейной (плоское) или билинейной интерполяции яркости по поверхностям граней (методы Гуро и Фонга)

Игровой пульт (геймпад – англ. gamepad, джойпад – англ. joypad) – тип игрового манипулятора. Представляет собой пульт, который удерживается

двумя руками, для управления используются большие пальцы рук (в современных геймпадах также часто используются указательные и средние пальцы).

Имитация (англ. simulation) – синоним термина «моделирование», но с подобием, устанавливаемым между объектом и его моделью не по физической сути, а по внешним признакам и производимым эффектам. В английской технической литературе И. обозначается термином «симуляция» (simulation). Использование слова «симуляция» в русскоязычной литературе вряд ли можно считать удачным, так как оно имеет давно установившееся негативное значение.

Иммерсивность (англ. immersibility) – одно из основных свойств системы виртуальной реальности. Благодаря И. система виртуальной реальности обеспечивает иммерсию, погружение пользователя в мир виртуальной реальности. Можно отметить три формы иммерсии: прямую, когда пользователь чувствует себя частью виртуального мира; опосредованную, когда участник видит себя или часть своего тела как бы со стороны и зеркальную, когда пользователь видит себя и виртуальный мир как бы в зеркале. Иммерсия (англ. immersion) – процесс погружения пользователя VR-системы в мир виртуальной реальности. В результате пользователь чувствует себя частью этого мира.

Можно встретить понятит полной И. (когда пользователь воспринимает виртуальный мир, как реальный) и неполной И. (когда имеет место частичное погружение). Последнее возникает в некоторых VR-системах с ограниченными средствами воспроизведения соответственных имитаторов. Строго говоря, о виртуальной реальности может идти речь только в случае полной И.

Интерактивность (англ. interactivity) – одно из основных свойств виртуальной реальности и, соответственно, VR-систем. И. заключается в том, что пользователь, погруженный в виртуальный мир, может взаимодействовать с виртуальной средой и находящимися в ней объектами. И. может проявляться в разных формах: как собственное движение в мире виртуальной

реальности, как воздействие пользователя на объекты виртуального мира и как обратное действие объектов виртуального мира на пользователя. Некоторые авторы отмечают, что понятие И. может включать в себя не только физическое, но и интеллектуальное взаимодействие.

Интерфейс (англ. interface) – техническое устройство, представляющее собой совокупность аппаратных и программных средств, обеспечивающих взаимодействие разнородных по виду энергии носителя и форме представления информации функциональных элементов, входящих в некую систему. В компьютерных и т.п. системах широко используются И., которые согласуют между собой электрические звенья с различными режимами и параметрами сигналов. В эргатических (человеко-машинных) системах, в том числе, и в системах виртуальной реальности особое значение имеют так называемые пользовательские И., обеспечивающие согласование «машин ной» части с человеком-пользователем. В системе виртуальной реальности таковыми являются И. «вы числительный комплекс–человек», включающий в себя визуальную и акустическую систему головного дисплея, подвижную платформу или кабину, и И. «человек–вычислительный комплекс», включающий в себя органы управления, управляющую перчатку, системы трекинга и др.

Искусственная реальность (англ. artificial reality) – термин, используемый М. Крюгером, для обозначения систем, в которых человек-пользователь взаимодействует с виртуальными объектами, но без полного погружения в виртуальный мир. В частности, М. Крюгером им предложены такие системы И.р., как Videodesk (видеостол) и Videoplace (видеоместо). С его точки зрения виртуальная реальность – это И.р., которая возникает при использовании управляющих перчаток и головных дисплеев.

Использование некоторыми авторами И.р. как синонима термина «виртуальная реальность» нельзя считать корректным. Термин И.р. в этом случае не обладает необходимой полнотой, так как в нем отражено лишь происхождение виртуальной реальности, но не отражено ее проявление.

Искусственный интеллект (ИИ; англ. artificial intelligence, AI) – свойство интеллектуальных систем выполнять творческие функции, которые традиционно считаются прерогативой человека; наука и технология создания интеллектуальных машин, особенно интеллектуальных компьютерных программ.

Каркасная модель (англ. frame model) – результат одного из первых этапов дискретного синтеза изображений объектов виртуальной реальности. В К.м. объекты представляются в виде конструкций, состоящих из ребер многогранников. При воспроизведении К.м. напоминает объемную проволочную конструкцию. Поэтому К.м. иногда называют проволочной.

Квазиреальность (англ. quasi reality) – этот термин, дословно означающий «как бы реальность», иногда используется как синоним виртуальной реальности. Термин К. не обладает необходимой полнотой, так как учитывает только проявление виртуальной реальности, не затрагивая ее сущности.

Киберпространство (англ. cyberspace) – метафорическая абстракция, используемая в философии и в компьютерных технологиях и являющаяся виртуальной реальностью; второй мир как «внутри» компьютеров, так и «внутри» компьютерных сетей.

Компьютерная графика (англ. computer graphic) – в общем случае отображение на экране дисплея компьютера графической (не текстовой и не символьной) информации в виде рисунков, чертежей, графиков и т.д.

К.г. означает также научно-техническое направление, занимающееся обоснованием, проектированием и созданием графической информации компьютерными средствами. Имеет важное значение для VR-систем, так как в этих системах графическая информация является доминирующей. Компьютерная реальность (англ. computer reality) – реальность, в которой объекты виртуального мира представлены в виде кодов в памяти компью-

терной системы. Используемый некоторыми специалистами термин «компьютерная виртуальная реальность» некорректен. Ибо К.р. обладает свойством виртуальности, но не является виртуальной реальностью. К.р. может лишь порождать виртуальную реальность.

Конвергенция (англ. convergence) – фактор бинокулярного зрения, основанный на мышечных ощущениях, вызванных изменением угла сведения осей правого глаза и левого при рассматривании объектов разной удаленности. С уменьшением расстояния угол конвергенции увеличивается, а мышечные напряжения усиливаются.

Математическая модель (англ. mathematical model) – модель, являющаяся совокупностью математических формул или иных способов математического представления объектов и процессов виртуального мира. Составление М.м. является исходной процедурой для последующего компьютерного синтеза. В большинстве случаев при разработке М.м. используется процедура аппроксимации. При этом сложность М.м. определяется как сложностью объектов и процессов виртуального мира, так и принятыми методами математического описания.

Межзрачковое расстояние (англ. interpupility distance) – расстояние между центрами зрачков глаз человека.

Многогранник аппроксимирующий (англ. approximating polyhedron) – геометрическое тело, которое состоит из множества взаимно связанных граней и с помощью которого аппроксимируются криволинейные объемные объекты произвольной конфигурации.

Многоугольник аппроксимирующий (англ. approximating polygon) – плоская замкнутая геометрическая фигура, которая состоит из множества непересекающихся отрезков прямых линий и с помощью которой аппроксимируется участок криволинейной поверхности. В большинстве случаев М.а. – это треугольники, реже четырехугольники и пятиугольники.

Мультимедиа (англ. multimedia) – информационная технология, основанная на совместном использовании различных видов информации: графической, текстовой, акустической, видео с диалоговым управлением. Некоторыми авторами М. рассматривается как предтеча VR-систем.

Навигационная система (англ. navigation system) – подсистема системы виртуальной реальности, определяющая пространственное положение пользователя в виртуальном мире, а также обеспечивающая целенаправленное ведение его в этом мире по заданным маршрутам.

Настольная система виртуальной реальности (англ. desktop virtual reality system) – VVR-система, в которой воспроизводящая система представлена в настольном исполнении. В этом отношении Н.с.в.р. по своей структуре подобна традиционным диалоговым компьютерным системам. Отличается от них, главным образом, использованием стереоскопического воспроизведения изображений и наличием специальных органов управления с тремя и более степенями свободы.

Н.с.в.р. обычно представляет собой установленный на столе монитор, воспроизводящий последовательно или одновременно изображение стереопары. Пользователь рассматривает изображение с помощью очков-затворов или стереоскопа. Н.с.в.р. используется в тех случаях, когда виртуальные объемные объекты находятся в фиксированных направлениях, в пределах малых углов обзора и на небольших удалениях от пользователя.

Отсечение (англ. clipping) – процедура электрического синтеза изображений, заключающаяся в исключении из процесса обработки объектов или их частей, находящихся за пределами пирамиды видимости. Благодаря этому резко сокращаются объем перерабатываемой информации и требования к производительности вычислительного комплекса системы виртуальной реальности.

Ощущение (англ. sensation) – первичная (простейшая) реакция органов чувств человека на внешние раздражители (стимулы). Характер ощущения определяется типом органа чувств.

О. имеют две основные характеристики: качественную, оцениваемую разновидностями параметров (цветность – для зрения, тон – для слуха, тип запаха – для обоняния и др.), и количественную, проявляющуюся в изменениях интенсивности значений.

Панорамирование (англ. panoraming) – последовательный обзор пространства в пределах больших углов с помощью оптической или иной системы съема визуальной информации, обладающей малыми углами зрения. В частности, панорамирование может иметь место, если человек пользуется биноклем или перископом, а также в случае перископной VR-системы.

Перспектива (англ. perspective) – фактор пространственного восприятия объектов внешнего мира, заключающийся в изменении видимых геометрических и энергетических параметров объектов по мере их удаления от наблюдателя.

Различают линейную П., заключающуюся в сокращении видимых размеров объектов по мере их удаления; воздушную П., проявляющуюся в размывании и снижении контрастности объектов по мере их удаления, вызванную непрозрачностью атмосферы; и цветовую П., придающую удаленным объектам голубоватый оттенок (из-за спектральных свойств атмосферы).

Проективное преобразование (англ. projective transformation) – преобразование геометрических многообразий, инвариантом которого является прямолинейность отрезков. В результате при П.п. отрезки прямых сохраняют свою прямолинейность, хотя их длина и ориентация могут быть изменены, треугольник – остается треугольником, хотя длины сторон и углы его могут быть изменены.

Это очень важное свойство широко используется в компьютерной графике, в том числе и в VR-системах и тренажерах. Например, при аппроксимации кривой ломаной линией операции пересчета П.п. можно производить лишь над узловыми точками (вершинами). Отрезки же аппроксимирующей ломаной линии реконструируются после П.п. методом линейной интерполяции. То же относится и к многоугольникам, и к многогранникам, информация о которых также может быть сведена к совокупности узловых точек с последующей (после П.п.) реконструкцией ребер и граней.

Проекция (англ. projection) – вид геометрической конфигурации или системы конфигураций после реализации процедуры проецирования. Характер П. зависит как от вида исходной конфигурации, так и от способа проецирования. Термин П. часто понимается как проецирование.

Прототипирование (англ. prototyping) – одна из функций VR-системы, заключающаяся в создании виртуального прототипа тех или иных объектов и систем, предшествующего их созданию в реальном мире. Например, выбор вариантов архитектурной планировки города, расстановки мебели в помещении, музейных экспозиций может быть произведен в виртуальной среде на виртуальных прототипах.

Псевдореальность (англ. pseudoreality) – термин, дословно означающий ложную реальность, иногда используется как синоним виртуальной реальности. В этом отношении термин П. не обладает необходимой полнотой, так как учитывает только искусственный характер виртуальной реальности, не затрагивая ее сущности.

Распределенная система (англ. distributed system) – система, состоящая из множества взаимно связанных подсистем, разнесенных в пространстве, но функционирующих как одно целое. Как правило, распределенная система является многопользовательской.

Возможны различные схемы организации Р.с. Так, Р.с. может представлять совокупность однородных и равнозначных подсистем, а также может быть построена по принципу «центр и филиалы». Известны также Р.с. с функциональным распределением подсистем.

Реалистичность (англ. realistic) – означает степень близости искусственно созданных (синтезированных) изображений хорошо знакомым объектам физического мира. Показатель Р. в известной мере является интегральным, т. е. учитывает многие факторы: содержание, структуру, внешний вид, детали, окраску и т.д. Строгая количественная оценка Р. затруднена, и ее дают эвристично и сугубо качественно. Одним из вариантов показателя Р. является фотореалистичность.

Ребро (англ. edge) – линия пересечения двух граней. Элемент математической модели сложных поверхностей и многогранников. Совокупность Р. многогранника образует его каркасную модель.

Семпл, сэмпл (англ. sample – образец) – относительно небольшой оцифрованный звуковой фрагмент.

Симулятор (англ. simulator) – в индустрии развлечений компьютерная игра, основанная на имитации движения транспортного средства (чаще всего полета летательного аппарата). Обычно динамика управления чрезвычайно упрощена и имеет мало общего с реальными аппаратами-прототипами. Приобретаемые на С. навыки управления не могут быть перенесены на реальные аппараты, что отличает С. от тренажера.

Система трекинга (англ. tracking system) – в системах виртуальной реальности устройство, предназначенное для отслеживания перемещений пользователя и/или частей его тела в пространстве.

С.т. состоит из генератора опорного (магнитного, светового, ультразвукового и т.п.) поля, в котором находится пользователь, и системы трекеров, связанных с пользователем. С.т. вырабатывает сигналы, содержащие информацию о перемещении и ориентации пользователя и/ или частей его тела в пространстве, которые вводятся в вычислительный комплекс.

Стереофония (англ. stereophony) – двухканальная система воспроизведения объемного звука, учитывающая бинауральный эффект. Система звуковоспроизведения состоит из двух пространственно разнесенных источников звука (громкоговорителей). Объемный эффект воспроизводится лишь частично, ограничиваясь лишь поперечными перемещениями между источниками звука.

Стереоизображение, стереоскопическое изображение (от др.-греч. στερεός – объёмный, пространственный) – изображение, вызывающее иллюзию объёма, то есть ощущение рельефности и протяжённости в глубину за счёт особенностей бинокулярного зрения.

Текстура (англ. texture) – специальный узор, наносимый на поверхности граней для придания реалистичного вида синтезируемым объектам при минимальных аппаратных и программных затратах в вычислительном комплексе, например: кирпичная кладка стен, паркетный пол, волнующееся море, пшеничное поле и т.п. Формируются и вводятся сигналы текстуры в сигналы синтезируемых изображений после этапа геометрических преобразований.

Текстурирование (англ. texture mapping) – процедура вынесения текстуры на грани аппроксимирующих многогранников и управление этой текстурой в соответствии с изменением пространственного положения многогранника относительно источника света и наблюдателя.

Технология виртуальной реальности (англ. virtual reality technology) – совокупность процедур, связанных с проектированием, созданием и использованием средств виртуальной реальности.

Трекер (англ. trecker) – в системах виртуальной реальности элемент системы слежения, связанный с пользователем. В частности, Т. могут устанавливаться на голове, руках и других частях тела пользователя. Головной Т. позволяет отслеживать повороты головы пользователя в пространстве, а ручной Т. – перемещения руки и т.д.

Тренажер (англ. trainer, Simulator) – техническое устройство, предназначенное для обучения операторов навыкам работы с тем или иным оборудованием. Т. является модельной системой, имитирующей работу реального оборудования.

Чаще всего Т. строят с использованием принципов аугментированной реальности. Пользователь, находящийся в Т., воспринимает интерьер помещения, кабины, органы управления, приборы и индикаторы как существующую реальность, а внешнюю (внекабинную) визуальную обстановку, звуковые эффекты и другие воздействия как виртуальную.

Задачи, решаемые на Т., весьма разнообразны. Это и первоначальное обучение пользователя навыкам управления оборудованием, и поддержание этих навыков при длительных перерывах в работе с оборудованием, переучивание пользователя на новом родственном оборудовании, тестирование пользователей на профессиональную пригодность и т.д. Использование Т. для решения этих задач эффективно, экономично и безопасно, чем и обусловлено широкое применение.

В Т. обычно предусмотрены средства для отработки навыков управления в экстремальных ситуациях: в сложных погодных условиях, при отказах тех или иных рабочих систем и т.п. Эта особенность Т. очень важна, так как обучение в реальных условиях небезопасно, а нередко и невозможно. Когда экстремальная ситуация встречается в реальных условиях, пользователь ввиду дефицита времени должен действовать автоматически и единственно правильным путем. А этого можно добиться лишь обучением на Т.

В Т. обычно предусмотрена подсистема управления и контроля процесса обучения, называемая пультом инструктора (учителя и т.п.). С этого пульта инструктор может вводить в процесс обучения нештатные ситуации, контролировать и оценивать действия обучающегося.

Области применения Т. весьма разнообразны. Наиболее обширна группа Т. транспортных средств. Это космические, авиационные, автомобильные, железнодорожные, трамвайные, велосипедные и др. Т.

По количеству обслуживаемых пользователей различают индивидуальные и групповые Т.

По объему выполняемых функций Т. подразделяют на процедурные (в которых отрабатываются навыки, определяющие, в основном, последовательность действий при выполнении той или иной процедуры управления), специализированные (для приобретения навыков работы с отдельными подсистемами более сложной системы), комплексные (в которых воспроизводятся в полном объеме все процедуры по управлению той или иной системой для всего персонала, участвующего в управлении этой системой) и др.

Представляют собой конструкции, надеваемые на голову и снабженные видеоэкраном и акустической системой.

Трёхмерная графика – раздел компьютерной графики, посвящённый методам создания изображений или видео путём моделирования объектов в трёх измерениях.

Физическая реальность или физический мир (англ. physical reality) – вид объективной реальности, учитывающий физические свойства объектов и процессов, существующих в природе вне прямой связи с восприятием их человеком.

Физическое моделирование (англ. physical modeling) – моделирование, основанное на принципах физического подобия.

Человеко-машинная система (англ. man-machine system) – система, состоящая из человеческих и машинных (технических) подсистем, функционирующих совместно и имеющих единое целевое назначение.

Типичными примерами человеко-машинных систем являются «летчик и самолет», «летчик и самолетный тренажер», «пользователь и компьютер», «человек и система виртуальной реальности».

Синонимом Ч.-м.с. является термин «эргатическая система». В рамках Ч.-м.с. человеческое звено с точки зрения психофизики рассматривается состоящим из трех элементов: сенсорного поля, системы принятия решения и моторного поля.

Шлем виртуальной реальности и очки виртуальной реальности – устройства, позволяющие частично погрузиться в мир виртуальной реальности, создающие зрительный и акустический эффект присутствия в заданном управляющим устройством (компьютером) пространстве.

Эффект присутствия (англ. presence effect) – эффект, возникающий в VR-системе или тренажере, заключающийся в том, что пользователь чувствует себя находящимся в виртуальном мире. Э.п. близок термину «иммерсия». Отличие заключается в том, что термин Э.п. отражает факт пассивного нахождения в виртуальном мире.

Учебное издание

Наталья Борисовна Паршукова

ВИРТУАЛЬНАЯ РЕАЛЬНОСТЬ

УЧЕБНОЕ ПОСОБИЕ

ISBN 978-5-907611-48-1

Работа рекомендована РИС ЮУрГГПУ

Протокол № 26, 2022

Редактор Л.Н. Корнилова

Технический редактор Н.А. Усова

Издательство ЮУрГГПУ

454080, г. Челябинск, пр. Ленина, 69

Подписано в печать 7.09.2022

Объем 9,9 уч.-изд. л. (29,95 усл. п. л.)

Формат 60х84/8

Заказ Тираж 100 экз.

Отпечатано с готового оригинал-макета в типографии ЮУрГГПУ

454080, г. Челябинск, пр. Ленина, 69