Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

ОТЧЕТ

по преддипломной практике

Место прохождения практики: ООО «Зэт Поинт Апп»
г.Минск
Сроки прохождения практики: с 23.03.2020 по 19.04.2020

Руководитель практики от
предприятия:
Михневич М.В.

Выполнил студент гр. 653501
Глотов А.А.

Руководитель практики
от БГУИР:
ассистент кафедры информатики
Кукар Е.В.

Минск 2013

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Возможности человеческого мозга уникальны. Благодаря накоплению жизненного опыта, умению быстро анализировать поступающие потоки данных и наблюдению за окружающим миром, он может предсказывать многие ситуации и ожидать конкретного поведения объектов в реальном мире. Так, профессиональный баскетболист, совершая дальний бросок из трехочковой зоны, за доли секунды просчитывает какую силу надо приложить к мячу и в каком направлении его бросить, исходя из веса мяча, собственных физических возможностей и расстояния до баскетбольной корзины. Правильно просчитанный бросок позволяет ему получить заветные три очка для его команды. Но даже если человек не является профессиональный баскетболистом, а просто любит играть в мобильные игры на телефоне по дороге на работу или учебу, или же он увлекается спортивными мобильными симуляторами, где бросает виртуальный мяч в корзину, то ему очень важно, чтобы этот мяч летел по всем законам физики и игра максимально симулировала реальный бросок мяча.

Для получения положительного игрового опыта важно, чтобы физический движок игры корректно просчитывал все физические взаимодействия игровых объектов. Если мяч полетит не в ожидаемую сторону и будет вести себя не так, как мяч в реальном мире, то игрок будет разочарован и может просто удалить игру, а создатель данной игры потеряет пользователя и возможную прибыль. Поэтому в играх очень важна правильная симуляция физики объектов.

Одним из важных элементов игрового процесса также является построение траектории движения объекта. Если пользователь, играя в симулятор бильярда, загнает шар в лузу с отскоком от борта стола, то игра построит ему траекторию отскока шара от борта, что позволит шару правильно выбрать точку удара и достичь своей цели. Симулятор баскетбола построит параболическую траекторию движения мяча. Симулятор снайпера построит баллистическую траекторию движения пули. В некоторых играх пользователю надо показать как будет двигаться объект после большого числа столкновений с другими объектами.

На данный момент многие физические движки дают реалистичную симуляцию поведения объектов в игровом мире. В основу большинства положен расчет изменения положений объектов каждый фрейм игры за счёт векторов сил действующих на объекты, какими являются линейная скорость объекта, гравитация и другие воздействующие силы. Так, шар, брошенный

от уровня земли под определённым углом, с некоторой силой будет лететь по параболической траектории и в некоторый момент коснётся земли.

Построение траектории объекта накладывает некоторые ограничения. Так, если каждый фрейм симулировать игру на несколько десятков секунд вперед и сохранять данные о перемещениях объектов для того, чтобы показать игроку траекторию движения объектов, исходя из изначальных сил, действующих на эти объекты, то это повлечет за собой уменьшение производительности игры, возможные притормаживания для выполнения всех расчетов и получение негативного опыта у пользователя. Поэтому разработчикам необходимо искать другие, более оптимальные пути решения этой проблемы.

В данном дипломном проекте ставятся следующие задачи:

– рассмотреть существующие игровые физические движки;

– ознакомится с особенностями и возможностями игровых движков Unity и libGDX;

– разработать программный модуль для построения траектории движения в двумерном пространстве, используя игровые движки Unity и libGDX, провести сравнения, выделить положительные и отрицательные стороны каждого из движков;

– разработать программный модуль для построения траектории движения в трехмерном пространстве.

В результате получился программный модуль, написанный на языке C# с использованием среды разработки Unity, который можно использовать для построения траектории движения объектов в двумерном и трехмерном пространстве.

1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Траектория и динамика движения

Траектория материальной точки — линия в пространстве, по которой движется тело, представляющая собой множество точек, в которых находилась, находится или будет находиться материальная точка при своём перемещении в пространстве относительно выбранной системы отсчёта.

Возможно наблюдение траектории при неподвижности объекта, но при движении системы отсчета. Так, звёздное небо может послужить хорошей моделью инерциальной и неподвижной системы отсчёта. Однако при длительной экспозиции эти звёзды представляются движущимися по круговым траекториям.

Возможен и случай, когда тело явно движется, но траектория в проекции на плоскость наблюдения является одной неподвижной точкой. Это, например, случай летящей прямо в глаз наблюдателя пули или уходящего от него поезда.

Принято описывать траекторию материальной точки в наперед заданной системе координат при помощи радиус-вектора, направление, длина и начальная точка которого зависят от времени. При этом кривая, описываемая концом радиус-вектора в пространстве может быть представлена в виде сопряженных дуг различной кривизны, находящихся в общем случае в пересекающихся плоскостях. При этом кривизна каждой дуги определяется её радиусом кривизны, направленном к дуге из мгновенного центра поворота, находящегося в той же плоскости, что и сама дуга. При том прямая линия рассматривается как предельный случай кривой, радиус кривизны которой может считаться равным бесконечности. И потому траектория в общем случае может быть представлена как совокупность сопряженных дуг.

Существенно, что форма траектории зависит от системы отсчета, избранной для описания движения материальной точки. Так, прямолинейное равномерно ускоряющееся движение в одной инерциальной системе в общем случае будет параболическим в другой равномерно движущейся инерциальной системе отсчёта.

Действующие на материальную точку силы в этом понимании однозначно определяют форму траектории её движения (при известных начальных условиях). Обратное утверждение в общем случае не справедливо, поскольку одна и та же траектория может иметь место при различных комбинациях активных сил и реакций связи.

Динамика — раздел механики, в котором изучаются причины возникновения механического движения. Динамика оперирует такими понятиями, как масса, сила, импульс, момент импульса, энергия.

Также динамикой нередко называют, применительно к другим областям физики (например, к теории поля), ту часть рассматриваемой теории, которая более или менее прямо аналогична динамике в механике, противопоставляя обычно кинематике (к кинематике в таких теориях обычно относят, например, соотношения, получающиеся из преобразований величин при смене системы отсчёта).

Иногда слово динамика применяется в физике и не в описанном смысле, а в более общелитературном: для обозначения просто процессов, развивающихся во времени, зависимости от времени каких-то величин, не обязательно имея в виду конкретный механизм или причину этой зависимости.

Динамика, базирующаяся на законах Ньютона, называется классической динамикой. Классическая динамика описывает движения объектов со скоростями от долей миллиметров в секунду до километров в секунду.

Однако эти методы перестают быть справедливыми для движения объектов очень малых размеров (элементарные частицы) и при движениях со скоростями, близкими к скорости света. Такие движения подчиняются другим законам.

С помощью законов динамики изучается также движение сплошной среды, т. е. упруго и пластически деформируемых тел, жидкостей и газов.

В результате применения методов динамики к изучению движения конкретных объектов возник ряд специальных дисциплин: небесная механика, баллистика, динамика корабля, самолёта и т. п.

Эрнст Мах считал, что основы динамики были заложены Галилеем.

Исторически деление на прямую и обратную задачу динамики сложилось следующим образом.

Прямая задача динамики: по заданному характеру движения определить равнодействующую сил, действующих на тело. Обратная задача динамики: по заданным силам определить характер движения тела.

Обратная задача динамики — определение координат тела и его скорости в любой момент времени по известным начальным условиям и силам, действующим на тело. Для ее решения необходимо знать координаты и скорость тела в некоторый начальный момент времени и силу, действующую на тело в любой последующий момент времени.

Силы в механике зависят от координат и скоростей движения тела.

Для нахождения координат тела в любой момент времени необходимо по известным значениям сил, действующих на тело, и известной массе тела, согласно второму закону Ньютона, определить его ускорение, а затем последовательным интегрированием ускорения аналитическими или численными методами найти новое значение скорости тела, его перемещение и координаты. Обратную задачу механики часто приходится решать инженерам при проектировании машин и механизмов.

Например, при расчете траектории космического корабля на основе знания начальных условий и гравитационных сил, действующих на него со стороны планет, необходимо решить прямую задачу механики. Зная силу взаимодействия гребного винта с водой и силу сопротивления воды движению корпуса судна, можно определить, как будет двигаться судно, какую скорость оно может развить.

Существуют и задачи динамики смешанного типа, например, вычисление движения тел с наложенными на них связями. В таких случаях задача сводится не только к определению движения каждой материальной точки системы, но и к нахождению сил реакций связей

1.2 История развития игровых движков

Вместе с созданием первых игр программисты пришли к тому, что каждая игра содержит общие компоненты, даже несмотря на различие аппаратных платформ. А первые игры имели место на игровых автоматах размером с холодильник.

Общая для игр функциональность — графические решения, игровые механики, расчет физики и другое — стала выделяться в отдельные библиотеки, но, для того чтобы быть «игровым движком» было еще далеко. Во многом это было связано с серьезным различием программно-аппаратных платформ и неопределенности в самих играх. Ведь жанры и типы игр еще предстояло изобрести, при том, что многие первые игры были текстовыми. Собственно, именно для ранних адвенчур и платформеров и стали возникать игровые движки, особенно с развитием графики — хорошим примером можно назвать Adventure Game Interpreter (AGI). При разработке King’s Quest в далеком 1984 году, программисты Sierra On-Line столкнулись с неудобством низкоуровневой разработки столь сложной и перспективной по графике в те времена игры — и разработали набор решений, которым и стал AGI. Всего на нем было выпущено 14 различных игр за 5 лет на 7 различных платформах, поэтому понятие “кроссплатформенность” было

важным уже тогда.

Однако, движки того времени редко выходили за пределы изначальной компании-разработчика и, как правило, были достаточно узкоспециализированными под конкретный жанр игры.

Ситуация начала меняться в 1993-м году после выхода игры Doom от компании id Software. Хотя при ее разработке использовались наработки движка Wolfenstein 3D, с точки зрения возможностей и модульности в ней был совершен настоящий технологический прорыв. В то время видеопроцессоры были не способны эффективно работать с трехмерной графикой, поэтому Джон Кармак (ведущий программист движка) выполнял все необходимые математические вычисления, служащие для манипуляции с трехмерными объектами, светом, затенением, наложением текстур и прочего самостоятельно. В результате, изображение выглядело трехмерным, на самом деле таковым не являясь. Поэтому Doom engine (первая версия id Tech) был не истинно трехмерным, а псевдотрехмерным. Но важно то, что техническая составляющая этой игры задала стандарт для того, что могло называться игровым движком. А именно, движок Doom был модульным, представлял из себя набор подсистем, в нем каждый четко отделенный программный слой отвечал за обработку своей порции данных. В результате, использовать его для различных игр (Hexen, Heretic, Strife) и силами сторонних разработчиков (Raven Software и Rogue Entertainment) стало намного проще. Поэтому появление игровых движков относят к середине 90-х годов 20-го века, то есть тогда окончательно сформировалось определение игрового движка в современном смысле.

Игровой движок представляет своеобразную узкоспециализированную операционную систему, поскольку включает все модули последней. В него входят: система управления памятью, графическая подсистема, система ввода, аудио подсистема, искусственный интеллект, физическая подсистема, сетевая подсистема, редактор игровых уровней и другое. Кроме того ядро движка может предоставлять особый подход к работе с файлами – файловую (ресурсную) систему, а так же отличающиеся от основной операционной системы средства работы с многопоточностью. Современный игровые движки вдобавок включают интерпретатор скриптового языка, заточенного для описания игровой логики, а нередко и полностью визуальный ее редактор. Его использование позволяет абстрагироваться от описания низкоуровневых команд и инструкций, а сконцентрироваться на геймплее. На этом составляющие движок компоненты не ограничиваются, их может быть как больше, так и меньше.

Игровой движок в первую очередь создается в целях упрощения и ускорения разработки. Поэтому включает средства для создания игрового мира – level-моделинга, импорта объектов, текстурирования, загрузки и анимации персонажей, создания визуальных эффектов, настройки физики и прочего.

Второй значительной целью разработки движка является кроссплатформенность или платформонезависимость разрабатываемой игры. То есть возможность ее запуска с минимально возможными изменениями. Совсем без изменений на другой платформе осуществить запуск игры не удастся из-за аппаратных различий, в том числе: размеров экрана, средств и способов управления и др.

Развитие игровых движков происходит вместе или под влиянием развития аппаратных и программных платформ, вместе с появлением новых игровых жанров и изменениями вкусов пользователей. Коротко говоря, развитием игровой индустрии в целом.

В середине 90-х после появления видеопроцессоров, способных обрабатывать трехмерную графику стали появляться программные интерфейсы, упрощающие ее разработку. Вслед за кроссплатформенным OpenGL на сцену в составе DirectX вышел Direct3D для Windows. Эти 2 визуализатора на много лет вперед определили способы графического вывода в играх.

В 1996-м году вышла игра Quake на Quake Engine. Этот движок оказал колоссальное влияние на игровую индустрию. Дерево движков, поулчивших свою жизнь благодаря Quake Engine представлено на рисунке 1.1.

Почти до конца десятилетия на рынке промежуточного программного обеспечения для игр (другими словами, игровых движков) практически единолично ритм задавала id Software. Однако в 1998-м году компания Epic Games выпустила успешную игру Unreal на одноименном движке — с настоящим технологическим прорывом по уровню графики. Ведущим программистом движка стал основатель Epic Тим Суини. Тим наравне с Кармаком является наиболее значимой фигурой в истории движков игровой индустрии — и Unreal Engine в его 3 и 4 версиях очень популярен и сейчас. Год спустя от Epic вышла ставшая еще более популярной игра Unreal Tournament.

В это же самое время конкурирующая компания-разработчик – id Software выпустила мультиплеерную игру Quake 3 Arena (на движке id Tech 3), ровно как Unreal Tournament включающую сетевые баталии.

Эти две игры стали флагманами индустрии, определив ее развитие на годы вперед.

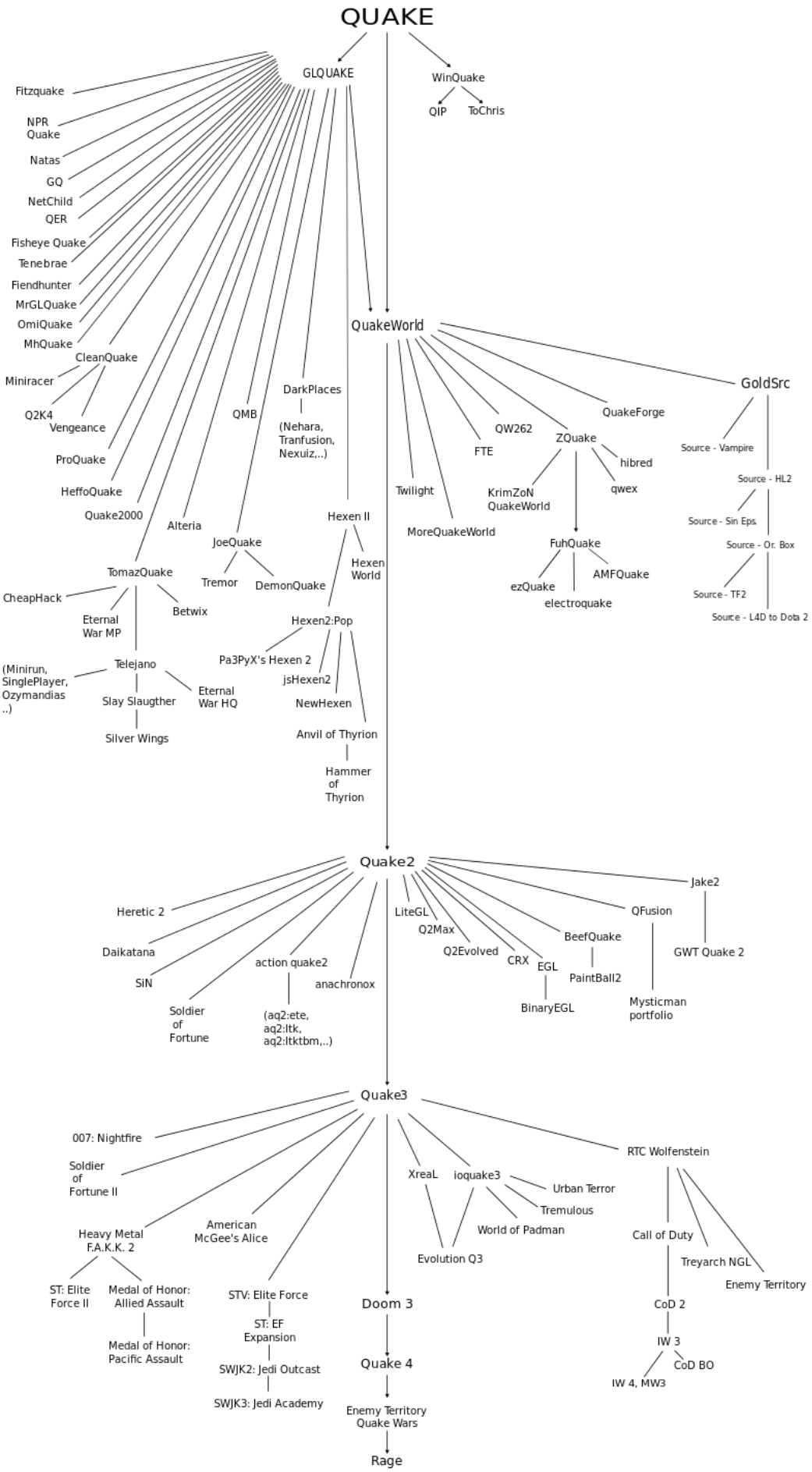

Рисунок 1.1 – Дерево движков, основанных на Quake Engine

На рынке было не так много игроков. Поэтому их продукция была очень дорога, и флагманские движки лицензировались только достаточно крупными разработчиками,

Ситуация начала коренным образом меняться примерно в середине первого десятилетия 21-го века. Тогда на рынке и в свободном доступе стало появляться большое количество средств для разработки игр. Бизнес промежуточного ПО (middleware) стал набирать обороты. Сначала рынок заполнился графическими фреймворками: Ogre, DarkGDK и др., предоставляющие программисту высокоуровневую прослойку над графическим API. В то же время отличающиеся от игровых движков полным отсутствием внутриигровых редакторов.

Затем на рынок пришли полноценные игровые движки по ценам, уместным для небольшой инди-команды разработчиков, среди них: Torque 3D, Unity 3D, и многие другие. Даже стартовавшие как флагманские движки — например, CryEngine от Crytek и ранее упомянутый Unreal Engine — стали использовать намного более доступную ценовую политику и стали доступны даже начинающим разработчикам.

Важным трендом игровой индустрии стали казуальные игры. Эти, по своей сути, незамысловатые, но красочные, не требующие бешеного взаимодействия с клавиатурой и мышкой головоломки с технической точки зрения были проще трехмерных хардкорных шутеров, поэтому для их разработки не понадобилось сильной модификации универсальных движков. Но, зато, в индустрии появились новые игроки, такие как: Torque Game Builder, HGE и другие.

В это же время, благодаря World of Warcraft, в игровой индустрии стали очень популярны MMORPG — а параллельно многие жанры делали все большую ставку на мультиплеер. Целый ряд движков не смог предоставить пользователям новую функциональность для клиент-серверных приложений, поэтому они ушли в небытие. Другие движки были адаптированы для мультиплеерного мира путем разработки для них серверных решений, так для Unity 3D были разработаны Photon и SmartFox. Третий тип универсальных движков, изначально являясь клиент-серверным, не почувствовал изменений. К нему относится Torque 3D. Также на рынке появились новые движки, предназначенные для глобальных многопользовательских игр, например HeroEngine, BigWorld, объединяющие масштабируемое под тысячи игроков серверное решение и доступный конкретному игроку клиент.

На рынке еще с 90х существовали браузерные игры, а затем второе рождение им дали социальные сети. необходимость эффективно создавать

игры для браузера не осталась незамеченной. Разработчики универсальных движков, например Torque 2D/3D, Unity 3D отреагировали на это довольно оперативно, выпустив плагины для браузеров, которые позволили отображать графику прямо в окне последних. Сначала популярность завоевал визуализатор на основе технологии Flash, но по целому ряду причин эта технология все больше теряет свою долю на рынке. Поэтому сейчас для визуализации в вебе часто используется библиотека для языка JavaScript — WebGL, которая позволяет создавать интерактивную 3D-графику. Однако, из-за недостатков языка, таких как отсутствие многопоточности, библиотека не может полноценно удовлетворить потребности игроделов. Ей на смену консорциумом W3C (куда входят: Microsoft, Google, Mozilla и др.) разрабатывается новый низкоуровневый бинарный компилируемый формат WebAssembly.

Под конец первого десятилетия 21-го века очень быстро развивались мобильные технологии. Как гром среди ясного неба появились мобильные устройства по мощности сопоставимые с ПК средней ценовой категории и способные запускать мощные игровые приложения со всеми спецэффектами, которыми обладали низкоуровневые графические интерфейсы. На что разработчики игровых движков ответили в некоторых случаях созданием специализированных конверторов, создающих нативный для конкретного оборудования код (как, например, Unity 3D), а в других — модернизировали свои продукты для кроссплатформенности (к примеру, Torque 2D, Cocos 2DX). Также, на рынке появились новые игроки, предлагающие кроссплатформенные движки для всего парка мобильных устройств, выполняющиеся со скоростью нативного кода. Примеры подобных средств: Corona SDK, Marmalade SDK, AGK (App Game Kit).

Также, возник целый ряд кроссплатформенных движков, позволяющих разработать игру при минимальном знании программирования. Примерами можно назвать Construct 2 и GameMaker Pro. Используя готовые решения и визуальные редакторы, можно быстро — иногда в течение нескольких часов — создавать простые игры. Это оказалось особенно распространенным на мобильном рынке, где распространение free2play модели и короткая игровая сессия сделали “простые” игры вполне успешным жанром.

1.3 Популярные кроссплатформенные игровые движки

1.3.1 Unity Engine Межплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity

позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идет постоянное развитие.

Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и независимыми студиями.

1.3.2 Unreal Engine Игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Первой игрой на этом движке был шутер от первого лица Unreal, выпущенный в 1998 году. Хотя движок первоначально был предназначен для разработки шутеров от первого лица, его последующие версии успешно применялись в играх самых различных жанров, в том числе стелс-играх, файтингах и массовых многопользовательских ролевых онлайн-играх. В прошлом движок распространялся на условиях оплаты ежемесячной подписки; с 2015 года Unreal Engine бесплатен, но разработчики использующих его приложений обязаны перечислять 5% роялти от общемирового дохода с некоторыми условиями.

Написанный на языке C++, движок позволяет создавать игры для большинства операционных систем и платформ: Microsoft Windows, Linux, Mac OS и Mac OS X; консолей Xbox, Xbox 360, Xbox One, PlayStation 2, PlayStation 3, PlayStation 4, PSP, PS Vita, Wii, Dreamcast, GameCube и др., а также на различных портативных устройствах, например, устройствах Apple (iPad, iPhone), управляемых системой iOS и прочих. (Впервые работа с iOS была представлена в 2009 году, в 2010 году продемонстрирована работа движка на устройстве с системой webOS).

Для упрощения портирования движок использует модульную систему зависимых компонентов; поддерживает различные системы рендеринга (Direct3D, OpenGL, Pixomatic; в ранних версиях: Glide, S3, PowerVR), воспроизведения звука (EAX, OpenAL, DirectSound3D; ранее: A3D), средства голосового воспроизведения текста, распознавание речи[8][9][10], модули для работы с сетью и поддержки различных устройств ввода.

Для игры по сети поддерживаются технологии Windows Live, Xbox Live, GameSpy и прочие, включая до 64 игроков (клиентов) одновременно. Таким образом, движок адаптировали и для применения в играх жанра MMORPG (один из примеров: Lineage II).

1.3.3 Cocos2D Кросс-платформенный фреймворк, используемый для разработки интерактивных приложений и игр (преимущественно для мобильных устройств). Является открытым программным обеспечением. Cocos2d содержит множество ответвлений, таких как Cocos2d-ObjC, Cocos2d-x, Cocos2d-html5 и Cocos2d-XNA. Также в сообществе Cocos2d имеется несколько независимых редакторов, предназначенных для редактирования спрайтов, частиц, шрифтов и тайловых карт. Можно также упомянуть редакторы мира: CocosBuilder и CocoStudio.

Работа всех версий Cocos2D основана на использовании спрайтов. Спрайты можно рассматривать как простые 2D изображения, но также может быть контейнером для других спрайтов. В Cocos2D, расположенные вместе спрайты создают сцену, к примеру, уровень игры или главное меню. Спрайтами можно управлять на основе событий в исходном коде или как часть анимации. Над спрайтами можно проводить всевозможные действия: перемещать, поворачивать, масштабировать, изменять изображение и так далее.

Cocos2D обеспечивает базовые примитивы анимации, которые используют спрайты. Некоторые версии Cocos2D позволяют эффекты частиц и применение шейдерных фильтров (warp, ripple и тд.).

Cocos2D предоставляет примитивы для создания простых элементов графического интерфейса. Они включают в себя текстовые поля, надписи, меню, кнопки и другие распространённые элементы.

1.3.4 CryEngine Игровой движок, созданный немецкой частной компанией Crytek в 2002 году и первоначально используемый в шутере от первого лица Far Cry. «CryEngine» — коммерческий движок, который предлагается для лицензирования другим компаниям. С 30 марта 2006 года все права на движок принадлежат компании Ubisoft.

Движок был лицензирован компанией NCSoft для разрабатываемой MMORPG Aion: Tower of Eternity.

В конце сентября 2009 года братья Ерли, основатели Crytek, дали интервью великобританскому журналу Develop, в котором заявили, что изначально CryEngine не планировался для лицензирования сторонними компа-

ниями. CryEngine планировался стать закрытым движком для сугубо внутреннего использования.

Игровой движок CryEngine — первый коммерческий движок Crytek. Его разработка была начата сразу же после основания компании. Движок первоначально разрабатывался как технологическая демонстрация для американской компании nVidia. Однако на выставке ECTS 2000 (англ. European Computer Trade Show — Европейская Компьютерная Выставка) Crytek произвела большое впечатление на всех больших издателей, посетителей и журналистов их технической демонстрацией, которая была показана в отделе nVidia. После этого на основе движка было решено создать 2 игры — «X-Isle» и «Engalus». Ни одна из этих игр так и не была выпущена.

2 мая 2002 года Crytek официально объявляет о том, что их игровой движок CryEngine полностью закончен и готов для лицензирования сторонними компаниями. Crytek также предлагает для лицензирования свою новую разработку — программу PolyBump.

26 марта 2004 года первая коммерческая компьютерная игра от Crytek и первая игра, использующая CryEngine — «Far Cry» – отправилась к розничным продавцам.

Особенности:

– CryEngine Sandbox: редактор игры в реальном времени, предлагающий обратную связь «Что Вы видите, то Вы и ИГРАЕТЕ».

– Рендерер: интегрированные открытые (англ. outdoor) и закрытые (англ. indoor) локации без швов. Также рендерер поддерживает OpenGL и DirectX 8/9, Xbox с использованием последних аппаратных особенностей, PS2 и GameCube, а также Xbox 360.

– Физическая система: поддерживает инверсную кинематику персонажей, транспортные средства, твёрдые тела, жидкость, тряпичные куклы (англ. rag doll), имитацию ткани и эффекты мягкого тела. Система объединена с игрой и инструментами.

– Инверсная кинематика персонажей и смешанная анимация: позволяет модели иметь множественные анимации для лучшей реалистичности.

– Система игрового искусственного интеллекта: включает командный интеллект и интеллект, определяемый скриптами. Возможность создания особенных врагов и их поведения, не касаясь кода C++.

– Интерактивная динамическая система музыки: музыкальные дорожки отвечают действиям игрока и ситуации и предлагают качество CD-диска с полным 5.1 звуковым окружением.

– Звуковое окружение и механизм SFS: способность точно воспро-

извести звуки от природы с плавным сопряжением без шва между средами и внутренними/внешними местоположениями в системе Dolby Digital 5.1. аудио. Включает аудио поддержку EAX 2.0.

– Сетевая система «клиент-сервер»: Управляет всеми сетевыми подключениями для режима с несколькими игроками. Это — система сети с низким временем отклика, основанная на архитектуре клиент-сервер.

– Шейдеры: скриптовая система используется для комбинирования текстур по-разному для увеличения визуальных эффектов. Поддерживается реальное попиксельное освещение, ухабистые отражения, преломления, объёмные эффекты жара, анимированные текстуры, прозрачные компьютерные дисплеи, окна, пулевые отверстия, и некоторые другие эффекты.

– Ландшафт: Используется расширенная карта высот и сокращение полигонов для создания массивной, реалистической среды. Видимое расстояние может составить до 2 км, когда преобразовано из игровых модулей.

– Освещение и тени: комбинация предрасчётных теней и теней реального времени, стенсильные тени и lightmaps (карты теней) для улучшения динамического окружения. Включает правильную перспективу с высокой разрешающей способностью и объёмные гладко-теневые реализации для драматического и реалистического внутреннего затенения. Поддержки продвинутых технологий частиц и любой вид объёмных эффектов освещения на частицах.

– Туман: включает объёмный, слоистый и дальний туман для увеличения атмосферы и напряжения.

– Интеграция инструментальных средств: объекты и строения, которые созданы на 3ds Max или Maya, интегрированы в пределах игры и редактора.

– Технология PolyBump: Автономная или полностью интегрированная с другими инструментальными средствами, включая 3ds max.

– Скриптовая система: Базируется на популярном языке Lua. Эта удобная система позволяет установку и тонкую настройку параметров оружие/игра, проигрывание звуков и загрузку графики без использования кода C++.

– Модульность: Полностью написанный в модульном C++, с комментариями, документацией и разделами в множественных DLL-файлах.

– Geometry Instancing.