Петрозаводский государственный университет

Бетелев В.К.
Научный руководитель: Богоявленский Ю.А.

Разработка прототипа игры для мобильных устройств в 3D представлении для одного или двух игроков

72-я Всероссийская (с международным участием) научная конференция обучающихся и молодых ученых
26 ноября, 2020, Петрозаводск, Россия

Цель работы

Создание прототипа игры для мобильных устройств в 3D представлении для одного или двух игроков.

Задачи

- Определить особенности создания игры для нескольких игроков.
- Определить особенности перевода игрового процесса в цифровой геймплей.
- Разработать алгоритм игрвого контроллера, управляющего игровым процессом для одного или двух игроков.
- Разработать прототип игры с понятным польователю интерфесом и простым управлением.

Игра kykke

За основу для создания прототипа была взята игра kykke (на русском – кююккя).

Старинная народная командная игра, популярная у финнов и карелов, внешне напоминающая русскую игру в городки. По традиции игроки побеждённой команды катают победителей верхом на спине вокруг площадок.

Средства разработки

Для создания прототипа игры был выбран игровой движок Unity 3D версии 2019.3.1f1 по следующим причинам:

- Кроссплатформенность – возможность “в один клик” выпускать готовую игру под другую платформу (Android, IOS).
- Простота использования – благодаря огромному набору готовых функций и методов, разработка игр на Unity позволяет разработчику не отвлекаться на создание “велосипеда”, а сосредоточится на разработке интересного игрового процесса.
- Удобный скриптовый язык программирования C#.

Для создания текстур использовался растровый редактор GIMP.
Для создания 3D моделей использовалась программа Blender.

Реализация

Передвижение персонажей.

Перед стартом игры, персонаж игры разворачивается к игровому полю.
Когда игрок нажимает на одну из кнопок интерфейса Left или Right, первому или второму персонажу передаётся команда для смены позиции. Пока игрок продолжает нажимать на кнопку, персонаж передвигается в сторону до те пор, пока не достигнет крайней позиции, или игрок не перестанет нажимать на кнопку.

Реализация

Бросок биты.

Когда пользователь нажимает на игрового персонажа, объект “бита” перестаёт быть дочерним персонажу, становится самостоятельным объектом и ему придаётся ускорение в определённом направлении через метод AddForce с помощью компонента Rigidbody(стандартный компонент Unity), отвечающего за физику. Сила броска зависит от положения специального объекта SliderFrow, значение которого можно получить с помощью компонента Slider(стандартный компонент Unity) и его параметра value.

Реализация

Бросок биты.

Также бита закручивается с помощью метода AddTorque компонента Rigidbody(стандартный компонент Unity). В параметры броска как через AddForce, так и через AddTorque, вносятся каждый раз новые случайные значения для большей вариативности и увлекательности игрового процесса.

Камера игрока начинает двигаться по вектору движения биты, дабы проследить за полётом биты.

Реализация

Отслеживание попадания и падения кюкки (городка).

С помощью компонента MeshCollider(стандартный компонент Unity) получаем событие, что какой-то объект коснулся городка. Начинаем проверять угол наклона городка в глобальных координатах по двум осям к нулевым глобальным координатам. Если этот угол больше или равен 135, то городок считается упавшим. (135 градусов – игровая условность).

Если играют два игрока, то один балл начисляется тому, чей городок был сбит. Если сбит чужой городок, то у игрока снимается один балл.

Реализация

Накопление баллов и победа в игре.

Когда городок упал, вызывается метод AddPoints и компонента Cube(создан вручную). Если играют два игрока, то один балл начисляется тому, чей городок был сбит. Если сбит чужой городок, то у игрока снимается один балл.

Вместе с созданием уровня – инсталированием на сцену кубиков в определённом порядке – передаётся и количество очков для победы в переменную countPointsToWin компонента GameController(создан вручную). После каждого обновления очков у обоих игроков вызывается метод CheckIfWin компонента GameController(создан вручную), проверяющий, было ли одним из игроков накоплено количество очков, необходимых для победы.

Оптимизация

Так как игра создавалась для мобильны платформ, то были подробно изучены и применены следующие варианты оптимизации.

- Объединение текстур в текстурные атласы. Видеоядро теперь загружает не много текстур по отдельности, непрерывно обращаясь к памяти устройства, а одну большую текстуру. Прирост в скорости загрузки был отмечен в 2,5 раза.
- Оптимизация полигональной сетки моделей и использование подхода low-poly, дабы сократить кол-во вызовов отрисовки полигонов видеоядром.
- Использование Occlusion Culling – системы, с помощью которой приложение получает информацию о тех объектах, которые видит или не видит камера игрока. Тогда можно отрисовывать исключительно объекты, попадающие в поле видимости игрока и не тратить ресурсы на отрисовку объектов, которых игрок не видит.

Оптимизация

Также был оптимизирован и программный код игры.

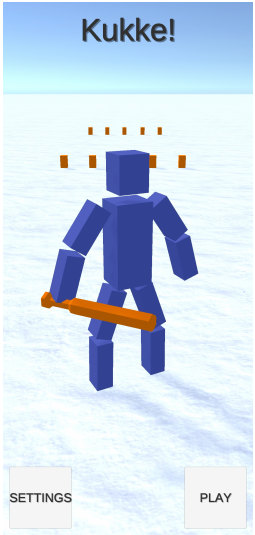
Стандартный метод Update, который сам движок вызывает каждый кадр, удалось уменьшить до 5 строчек. Все физические вычисления были перенесены в специальный метод FixedUpdate, который не зависит от количества кадров в секунду (FPS – frame rate per seconds), а подчиняется внутреннему измерению времени. То есть, даже если у игрока на устройстве будет малый FPS, все физические вычисления (полёт биты, падение городка) будут просчитываться верно и вовремя.

Игровая хитрость

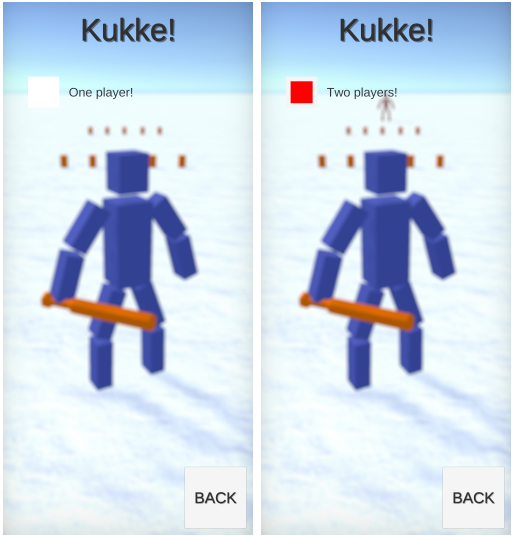
Для усложнения игрового процесса был добавлен специальный слайдер, отвечающий за силу закручивания биты при броске.

Однако на самом деле этот слайдер никуда не отправляет данные и является лишь “заглушкой”, чтобы создать у пользователя ощущения сложности игровой механики и заставить его прочувствовать геймплей, увлечься тактикой и расчётами броска.

Стартовый экран

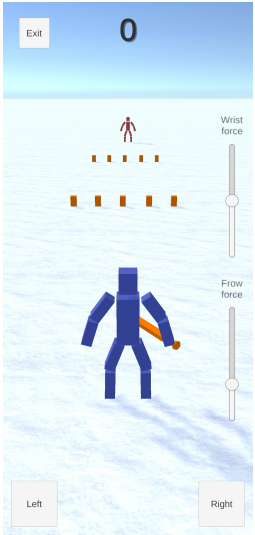


Экран выбора режима игры



Преигровой экран

Главный экран игры



Заключение

Все поставленные задачи были выполнены, данный прототип полностью готов для дальнейшего развития. Например:

- Можно заменить модели персонажей;
- Создать много новых уровней с разной расстановкой городков и разным количеством победных баллов;
- Заменить текстуры, добавить спецэффекты;

Спасибо за внимание!