Санкт-Петербургский государственный университет

Технологии программирования

Группа 21.Б07-мм

Компьютерная игра в жанре «выживание» на Unity Engine

Колесников Михаил Олегович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
доцент кафедры системного программирования СПБГУ, к.т.н. Ю.В. Литвинов

Санкт-Петербург
2023

Оглавление

Введение

Сейчас компьютерные игры становятся все более популярными. Одним из известных жанров игр является выживание с открытым миром [5]. Одними из игр с большой игровой аудиторией такого жанра являются: Minecraft, The Forest, Raft. Главной особенностью таких игр является то, что игроку нужно постоянно следить за своими потребностями, чтобы выжить и не умереть. Для этого ему нужно будет добывать еду, делать постройки, находить новые ресурсы и другие предметы. Игры про выживание создают испытания для игрока — выжить, минуя смерть. Именно это и делает их столь интересными для пользователей.

Для написания 3D игр используются различные игровые движки. Unity Engine — кроссплатформенный игровой движок, который поддерживается macOS, Windows и Linux. Он позволяет писать игры не только на ПК, но и на консоли и мобильные устройства.

Было решено создать 3D игру с низкополигональной графикой с видом от первого лица. Основной идеей игрового пространства является процедурная генерация пещер, которые заполняют весь мир. Важной их особенностью является разрушаемость. Каждая пещера будет иметь собственный биом, в котором будут животные, растения или ресурсы определенного типа. Усложнением игры являются различные монстры, заполняющие укромные уголки. Главной задачей игрока станут нахождение ресурсов для крафта и строительства, предназначенного для создания убежища или красивых построек. Вид от первого лица позволит сильнее погрузиться в атмосферу замкнутого пространства. В течение всего выживания игрока будут поджидать внезапные землетрясения, сопровождаемые обрушениями, наносящими урон.

Дата сборки: 5 января 2023 г.

1. Постановка задачи

Целью данной работы является создание приложения на ПК — компьютерной игры про выживание в мире, состоящем только из пещер. Для её выполнения были поставлены следующие задачи.
Осенний семестр:

1. Выполнить обзор аналогов.
2. Спроектировать архитектуру проекта.
3. Реализовать пользовательский интерфейс игрока.
4. Реализовать генерацию мира.

Весенний семестр:

1. Реализовать создание предметов из добываемых игроком ресурсов.
2. Реализовать механику строительства.
3. Реализовать разделение мира на биомы с различными особенностями.
4. Провести апробацию.

2. Обзор

2.1. Обзор аналогов

Для сравнения выбраны игры про выживание, а именно: Raft, The Forest, Minecraft. Целью обзора является выявление механик, присущих играм такого жанра.

Raft. Игра, написанная на игровом движке Unity Engine, где землю полностью затопило водой, главному герою предстоит выжить в этом мире. Игрок самостоятельно строит плот из собранного мусора, испытывая постоянные атаки акулы. Особенной механикой является разделение игры на этапы, в зависимости от которых игроку открываются новые рецепты для создания предметов. Особенностью генерации мира является прорисовка заранее заготовленных островов вокруг игрока. Игра порождает случайные острова с различными ресурсами. Из-за такой генерации игрок может встретиться с экземпляром одного и того же острова, но с новыми ресурсами. На тип островов также влияет прогресс игрока.

The Forest. Игра, разработанная на Unity Engine, посвящена выживанию на острове после крушения самолёта. Игроку необходимо найти пропавшего сына, но сложностью являются туземцы-каннибалы, населяющие этот остров. Интересной механикой является строительство. В игре можно создавать такие объекты, как фундамент, убежища, костры и многое другое. Все постройки могут быть сломаны врагами, но игрок может их починить с помощью инструментов. Особенностью игры являются возобновляемые ресурсы. После перезапуска игры подобранные игроком вещи будут возобновлены, а также при повторном спуске в пещеры все вещи, находящиеся в ней тоже обновятся.

Minecraft. Цель игры — выжить в бескрайнем мире, состоящем из различных биомов. Осложняет это наличие голода и монстров, выползающих по ночам. Основной механикой этой игры является мир с процедурной генерацией. Весь мир поделен на чанки (от англ. «chunk» — «кусок»), состоящие из блоков 16 в ширину, 16 в длину и 384 в высоту.

А также весь мир поделен на различные биомы с разными противниками и ресурсами. Карта биомов генерируется за счет карт влажности и температуры, которые создаются с помощью шума Перлина [6]. В отличие от вышеупомянутых игр, Minecraft написан не на игровом движке, а на Java [3] с использованием графической библиотеки LWJGL [4].

2.2. Существующие решения

2.3. Используемые технологии

Среди игровых движков можно выделить Unity Engine и Unreal Engine. Для разработки игры был выбран Unity Engine, потому что он является более легким для изучения начинающим разработчиком. Основным классом в Unity Engine является базовый класс MonoBehavior, от которого наследуются все скрипты.

Основным языком программирования выбран C#, поскольку этот язык используется для написания скриптов в Unity Engine.

Для реализации шаблона проектирования «Сущность Компонент Система» (Entity Component System) [2] используют различные фреймворки, а именно: Entitas, LeoEcsLite, Morpeh. Для разработки был выбран LeoEcsLite, так как он имеет хорошую производительность, является минималистичным с простым API, а также имеет большое комьюнити. Этот фреймворк написан на C# и имеет открытый исходный код. Сущность является контейнером для компонентов. Компонент — контейнер для данных, который не должен содержать логику. Каждая система, существующая в виде пользовательского класса, должна реализовывать хотя бы один из IEcsInitSystem, IEcsDestroySystem, IEcsRunSystem интерфейсов, предлагаемых фреймворком. Для хранения всех сущностей используется тип EcsWorld, являющийся контейнером для всех сущностей. Для включения всех систем используется специальный тип EcsSystems, являющийся контейнером для систем, которыми будет обрабатываться экземпляр мира EcsWorld.

Для интеграции LeoEcsLite с Unity Inspector используется расши-

рение UniLeo. Для работы с данными компонентов в Unity Inspector UniLeo позволяет создавать провайдеры MonoBehavior для компонентов. Провайдер представляет собой класс без логики, который наследуется от MonoProvider<T>, где T — любой компонент.

3. Реализация

3.1. Архитектура

Для проектирования архитектуры был выбран шаблон проектирования «Сущность Компонент Система». При таком подходе данные и логика разделены. Сущность (Entity) представляет собой набор ссылок на различные компоненты (Components) в то время как системы (Systems) содержат логику, взаимодействующую с данными компонентов. Точкой запуска игры является класс EcsGameStartUp, наследуемый от класса MonoBehaviour. В методе Start происходит инициализация полей world типа EcsWorld и systems типа EcsSystem. Далее происходит добавление всех созданных систем в systems. В методе Update, вызываемом каждый кадр, происходит поочередный запуск всех систем.

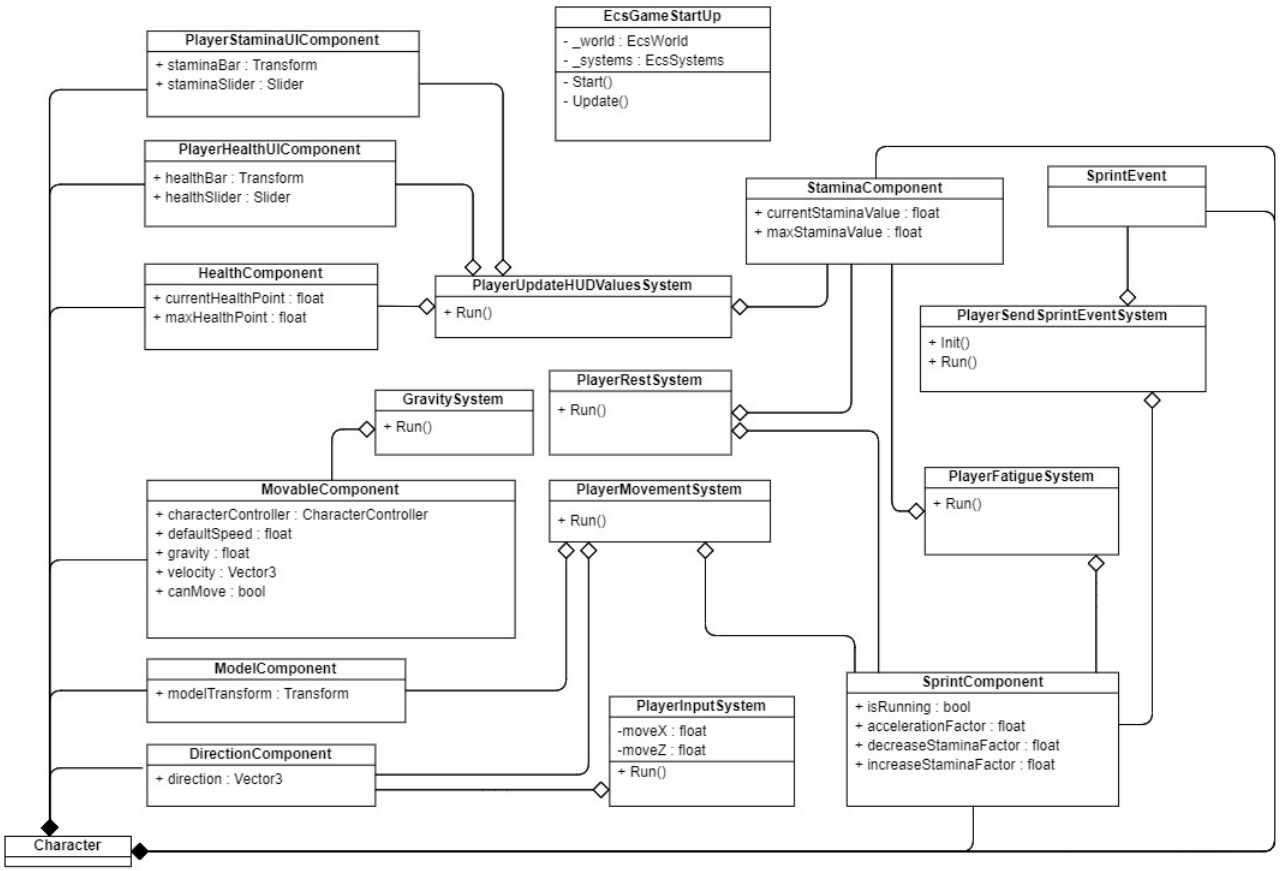

Рис. 1: Архитектура приложения (Часть 1)

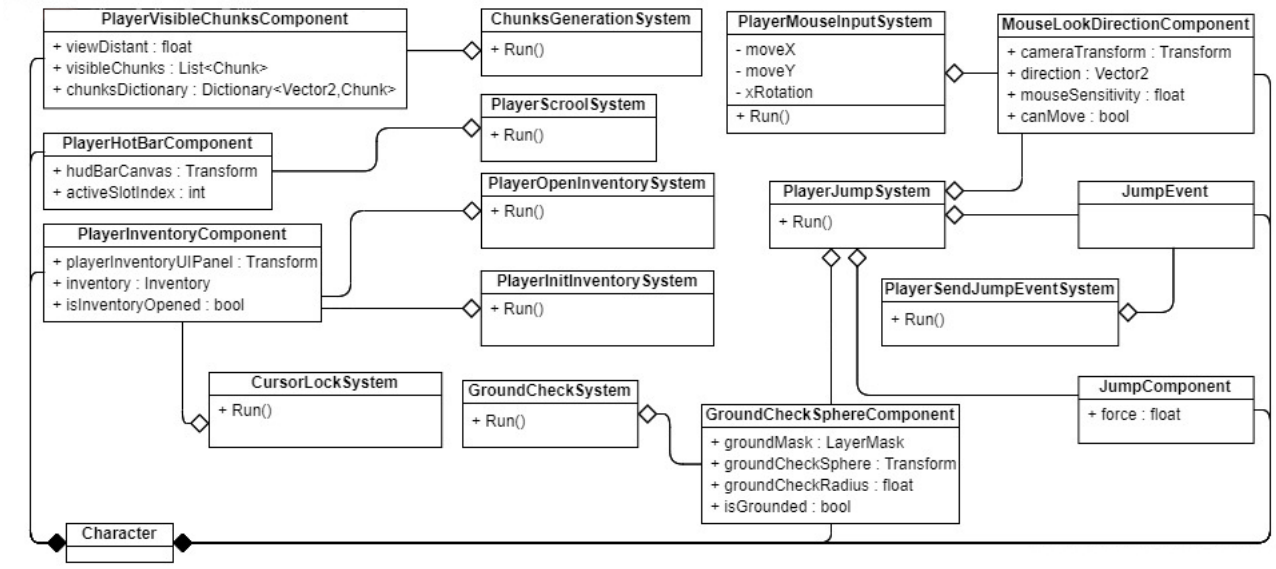

Рис. 2: Архитектура приложения (Часть 2)

3.2. Игровой персонаж

Для создания сущности персонажа, была создана простая 3D модель на сцене, чтобы добавлять на неё компоненты. Для того чтобы игрок мог взаимодействовать с игровым персонажем, через Unity Inspector на модель были добавлены компоненты провайдеры для каждого компонента, который имеет персонаж. Для реализации событий игрока, такие как прыжок и бег, созданы специальные компоненты без логики JumpEvent и SprintEvent, которые добавляются к сущности игрока системой PlayerSendJumpEventSystem и PlayerSendSprintEventSystem. В зависимости от того, есть ли на сущности игрока эти компоненты, выполняется логика в других системах. В конце выполнения всех систем в одном кадре со всех сущностей удаляется компоненты JumpEvent и SprintEvent.

Интерфейс игрока представляет собой возможность открывать инвентарь и отображение текущего количества здоровья и выносливости. Для реализации характеристик игрока были созданы компоненты HealthComponent и StaminaComponent. Так как сущность персонажа создается из инспектора, для этих компонентов реализованы классы HealthProvider и StaminaProvider. Для того чтобы на экране игрок ви-

дел здоровье и выносливость, созданы классы PlayerHealthUIProvider и PlayerStaminaUIProvider, хранящие поля слайдера и холста, представляющего собой абстрактное пространство для отрисовки UI.

3.3. Генерация мира

Для построения открытого мира было решено разделить пространство на отдельные чанки (от англ. «chunk» — «кусок») размером 8 в длину и ширину. Чанк представляет собой gameObject, который имеет коллайдер (от англ. «collide» — «сталкиваться») и меш (от англ. «mesh» — «ячейка» сетки). Генерация мира происходит в поле зрения игрока, выражаемого в количестве чанков, которые прогружаются рядом с игроком. Создание чанка представляет собой генерацию набора вершин и треугольников, определяющих поверхность. Для их создания был реализован алгоритм «Марширующие кубы» [1]. Логика обновления чанков представлена в системе ChunksGenerationSystem. Для создания меша чанка используется алгоритм марширующих кубов, представленный в классе MarchingCubes.

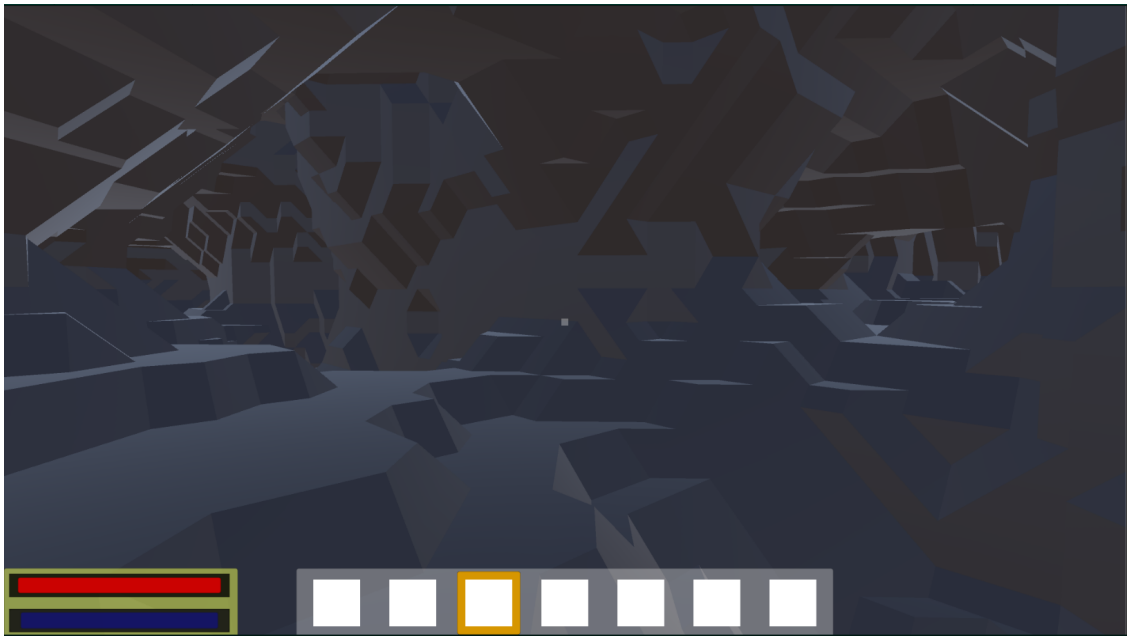

Рис. 3: Кадр из игры

Заключение

В итоге достигнуты следующие результаты

- Выполнен обзор аналогов;
- Спроектирована архитектура;
- реализован пользовательский интерфейс игрока;
- реализована генерация мира.

Ссылка на исходный код — https://github.com/L-glance/TheCaves

Список литературы

[1] Geiss Ryan // Generating Complex Procedural Terrains Using the GPU. — URL: https://developer.nvidia.com/gpugems/gpugems3/part-i-geometry/chapter-1-generating-complex-procedural-terrains-using-gpu (дата обращения: 2022-12-01).

[2] Riady Yos. ECS // Entity Component Systems in Elixir. — URL: https://yos.io/2016/09/17/entity-component-systems/ (дата обращения: 2022-12-01).

[3] Wikipedia. Java // Wikipedia, the free encyclopedia. — URL: https://ru.wikipedia.org/wiki/Java (дата обращения: 2022-12-26).

[4] Wikipedia. Lightweight Java Game Library // Wikipedia, the free encyclopedia. — URL: https://ru.wikipedia.org/wiki/Lightweight_Java_Game_Library (дата обращения: 2022-12-26).

[5] Wikipedia. Open World // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Open_world (дата обращения: 2022-12-01).

[6] Wikipedia. Perlin noise // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Perlin_noise (дата обращения: 2022-11-11).