LaTeX

Смельцова Ксения Валентиновна

25.12.2023

Содержание

1 Герой, Анимация и Текстуры

Создаем 2D Scene сцену. Это будет наш старт, небольшая заготовка для дальнейшего развития нашей игры. Дадим ей временное название "World".

Следом создаем новую сцену, но на этот раз не 2D Scene, а CharacterBody2D. Для удобства дадим имя "Player".

Пока наш герой не способен что либо делать. Для того что бы он хотя бы начал двигаться или прыгать необходимо прописать скрипт. Для этого мы создаем его, а после вносим туда дальнейшие коррективы по мере создания игры.

Переходим на сцену нашего персонажа и добавляем туда CollisionShape2D. В правой части экрана находим раздел под названием Shape. С помощью этого раздела мы задаем форму нашего героя. Из заранее заготовленных материалов находим нужный нам спрайт и просто перетаскиваем его на сцену в диапазон выбранный с помощью Shape. В моем случае это лягушка.

Можно заметить что спрайт перекинулся не совсем четким. Это легко исправить в настройках Godot, а именно:

- Проект
- Настройки проектов
- Рендерринг
- Текстуры
- Фильтр текстур по умолчанию
- Nearest

Это означает что наша картинка при увеличении не будет размалеваться. Не будет применяться эффект сглаживания и спрайт получится более четким.

Переходим в World, создадим там временное статическое тело, для того что бы наш герой мог где ни будь стоять или обо что ни будь ударяться. Это надо для того что бы проверить функционал нашего героя. Основные текстуры мы будем добавлять чуть позже, пока разберемся с настройкой спрайта.

Создаем StaticBody2D, следом в этом жу узле CollisionShape2D. Далее по старой схеме выбираем форму и переносим ее на экран. После все так же в этом узле создаем ColorRect. Это позволит нам дать цвет нашей фигуре.

Если нет подготовленных декораций, то их вполне можно создать прямо в Godot, он даст абсолютно все необходимое для этого, ограниченное лишь вашей фантазией.

После сохранения и выстраивания нашей сцены, перекинем туда игрока. Теперь он и его действия будут отображаться там. Что бы наш герой не был деревянным надо добавить к нему пару анимаций. Сделаем мы это с

помощью AnimatedSprite2D. Это позволит нам добавить и настроить чистоту кадров анимации. Необходимо будет в редакторе анимаций внимательно произвести подсчет кадров в нашем спрайте. Дальше просто вносим коррективы в FPS и при необходимости добавляем другие анимации. В моем случае я использовала чистоту кадров в 20 FPS и 7 анимаций из которых в итоговой работе было задействовано 2-3.

На этом этапе уже можно запускать игру и проверять нашего героя на анимации. В случае каких то ошибок их можно будет устранить. Так же не забывает о прописании скрипта для героя, одних отредактированных анимаций будет мало. Тут необходим скрипт в стандарте с действиями героя.

Если проверка прошла удачно и все работает, то можно начинать переходить к окружению. А именно добавления текстур, различных платформ и кубов. Это разнообразит внешний вид игры, как уже говорилось ранее, если нет заготовленного материала, можно создать все в Gjdot. В сцене мира создает узел TileMap и в нем же в правом в верхнем углу создаем TileSet, туда переносим заранее подготовленные текстурки. Так же в той же стороне можно настроить остальные все необходимые параметры.

На главном экране они просто так отображаться не будут. С помощью TileMap мы можем настроить и выбрать конкретные текстурки. Обрезав их по своему усмотрению и сохранив для дальнейшей эксплуатации. Теперь после всей обработки мы можем зажать мышку на понравившейся нам текстурке и выбрать необходимый нам участок. После спокойно навестись на мир где можно начать рисовать и расставлять все по своему усмотрению. После этого можно удалить старую платформу для испытаний героя.

2 Дверь, Переход на новый уровень

Заранее готовим наш спрайт двери, в моем случае это будет портал.

Так же можно простыми действиями сразу добавить пару уровней для нашей игры. Достаточно просто дублировать с помощью правой кнопки мыши наш мир. Так можно продолжать сколько угодно раз, главное не забыть визуально изменить уровень так как он будет полностью скопирован с первого мира. Для дальнейшего удобства предлагаю заменить название сцены "World"на "Levels1..2.."в зависимости от того сколько уровней мы вообще хотим добавить.

Создадим новую сцену для нашего портала, это будет 2D Scene. Дадим ей сразу удобное для нас название, к примеру ExitDoor. Создание новой сцены было необходимо для дальнейшего облегчения процесса создания игры. Так мы сможем просто переносить сцену с порталом в необходимый нам уровень, нежели если мы в каждой сцене уровня будем создавать новый портал. Это было бы слишком затратно и неэффективно.

Выносим на сцену заготовленную картинку с порталом. Надо учитывать тот факт что размер нашего героя 32X32, поэтому подгоняем портал под эти параметры. Теперь необходимо добавить области которые будут срабатывать для перехода на следующую сцену. Добавляем Area2D, сюда мы добавляем в виде дочернего узла CollisionShape2D, где по старой схеме выбираем форму и в этот раз подгоняем ее под размер нашего портала. После всех визуальных коррективов незабываем добавить скрипт в котором пропишем все необходимое для дальнейшего функционала.

После написания скрипта и поправки визуала, можно перейти к самому тестированию двери что бы удостовериться в ее работоспособности. В случае чего мы всегда можем устранить свои ошибки. Переносим сцену портала в сцену 1 уровня и ставим там где нам бы того хотелось. Так же поступаем и со вторым уровнем. После можно запустить игру и проверить работу. Так же не забываем провести все необходимые сигналы для полного функционала.

3 Меню и Кнопки

Создаем новую сцену CanvasLayer, сразу сохраняем и дадим ей имя "Menu". Тут создаем узел Button. Это обычная кнопка, она понадобится нам для создания самого меню и перехода на уровни. С правой стороны увидим поле в котором можно дать название этой кнопке. Пока она не имеет никакого функционал, но в дальнейшем мы это исправим. Создадим несколько кнопок, дадим им характерные названия, такие как:

- Меню
- Настройки
- Рекорд
- Выход

Не забываем добавить скрипт, к нему так же проведем сигналы для кнопок. После уже в самом скрипте с помощью сигнаров пропишем все необходимое для функционала. После прописания всего необходимого, можно начать заниматься визуальной частью игры.

Добавим узел MarginContainer. Он поможем нам настроить наши кнопки, а если быть точнее то их расположение на экране. Внесем сюда же вспомогательбный дочерный узел VBoxContainer. Он поможем нам правильно расположить текст по вертикале.

Создаем новую сцену CanvasLayer, но на этот раз это будет выход из уровня. Создаем так же кнопку и необходимые вспомогательные узлы. Переносим нашу кнопу в верхний правй угол. Так же не забываем прописать скрипт и провести сигналы.

После всей работы переносим наш выход на каждый уровень. Теперь у игрока появится возможность покинуть данный уровень без полного закрытия игры.

4 Ловушка

Создаем новую 2D сцену и даем ей название Spikes. Это будут наши острые пики которые будут торчать из пола, но мы сможем при необходимости повернуть их для дальнейшего размещения.

Переносим заранее подготовленные шипы на экран и создаем новый узел Area2D и туда же дочерний узел CollisionShape2D. Выбираем форму и подгоняем ее под размер нашей ловушки. Это необходимо для того что бы игрок при соприкосновении с ловушкой погибал.

Не забываем про скрипт, с помощью Area2D добавляем сигнал с наш код. Вносим все необходимое и можно начать переносить сцену шипов на сцену наших уровней. Расставим их по желанию и перейдем к тестированию.

Что бы после смерти игрока мы не видели сам уровень, создадим сцену смерти CanvasLayer. Переименуем ее под "GameOver". В нее добавим все необходимые узлы и таймер. Он нам нужен будет для того что бы игрок долго не висел в поле конца игры. Учитывая что саму надпись мы сделали как кнопка, при желании можно будет не дожидаясь таймера начать игру заново. Скорректируем код и на этом перейдем к другой теме.

5 Арбузик

Как обычно создаем новую сцену и даем ей имя "Strawberry". Туда же добавим CollisionShape2D и по классике выбираем форму. После ищем наш спрайт и переносим его на экран подгоняя размер под форму. После создадим дочерний узел AnimatedSprite2D. Это позволит нам сделать небольшую покадровую анимацию для нашего арбузика. Так же создадим AudioStremPlayer и дадим название Sound. Это позволит нам добавлять звуки при поедании нашего арбузика.

После того как мы выбрали и перенесли необходимый нам звук, его можно всячески дополнительно настроить.

Прописываем необходимый скрипт и в сцене героя создадим еще один узел Area2D. Там же создадим CollisionShape2D и выбираем фигуру. Опять подгоним ее под героя и это условно будет область поглощения арбузика. Внесем коррективы в код, присоединим сигнал и можно разносить арбузик по всем уровням.

6 Очки и Рекорды

Создадим скрипт без сцены и назовем его global. Его суть в том, что созданные переменные тут, будут видны во всех других скриптах. Поэтому начнем прописывать необходимый код и подключать сигналы.

После написания всего необходимого создаем новую сцену CanvasLayer и сразу дадим ей имя "HallofFame". Это будет нашим меню рекордов, где будет соответствующая надпись. Так же не забудем добавить кнопку выхода в меню по тому же принципу что и до этого.

Для красивого размещения надписей не забудем добавить MarginContainer и VBoxContainer.

Дописываем код и идем запускать игру для проверки.

Находим сцену выхода из уровня и туда добавим счетчик, что бы игрок мог сразу отслеживать сколько он набрал и максимальное количество очков. По старой схеме внесем пару коррективов в скрипт.

После запуска видим, что переход в рекорды успешно работает и сам счетчик тоже. Даже после выхода из уровня и переходом в рекорды, мы наблюдаем рекорд игрока за все время.

При желании можно добавить к арбузику звук максимального рекорда. То есть когда игрок будет зарабатывать новый рекорд, будет небольшой звук.

7 Фон

Переходим к примеру в сцену с меню и добавляем туда новый узел ParallaxBackground. Это такой контейнер который высчитывает все движения слоев которые в него будут добавлены, сами же слои надо добавлять отдельно в дочерний узел под названием ParallaxLayer.

Берем готовое изображение и перетаскиваем его на экран. Выставляем границы как нам необходимо, не забываем перенести все в ParallaxLayer иначе наша картинка будет на первом плане и не будет считаться как фоновым объектом.

На этом все, проделываем тоже самое и со стальными сценами где нам необходим фон.

8 Вывод

По итогу мы имеем готовую игру за короткие сроки. В Godot можно создавать не только 2D платформер, но и 3D. Все необходимое для этого нам предоставили, осталось лишь правильно воспользоваться.

В моем случае я создала игру про лягушку. Ее целью является пройти несколько измерений что бы найти свой дом, по пути она сталкивается с преградами в виде ловушек. Но разве это ее сможет остановить? Так же она питается арбузиками которые предают ей силы для достижения своей цели.