Головной дисплей

Заключительный этап нашей игры нуждается в Пользовательском интерфейсе (UI), чтобы отображать такие вещи, как рейтинг, сообщение "игра окончена" и кнопку перезапуска.

Создайте новую сцену и добавьте узел CanvasLayer с именем `HUD`. «HUD» означает «heads-up display», информационный дисплей, который отображается в виде наложения поверх игрового вида.

Узел CanvasLayer позволяет нам прорисовывать элементы нашего UI на слое поверх всей остальной игры, поэтому отображаемая информация не перекрывается никакими игровыми элементами, такими как игрок или мобы.

HUD должен отображать следующую информацию:

- Счет, измененный `ScoreTimer`.
- Сообщение, например "Game Over" или "Get Ready!"
- Кнопка "Start", чтобы начать игру.

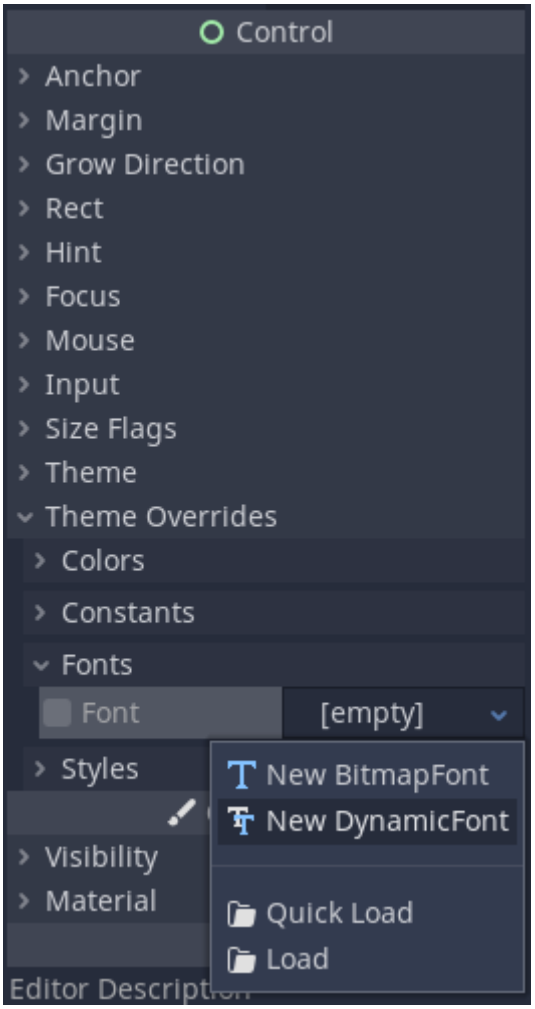
Основной узел для элементов UI — это Control. Чтобы создать наш UI, мы будем использовать два типа узлов Control: Label и Button.

Создайте следующие узлы в качестве потомков узла " HUD":

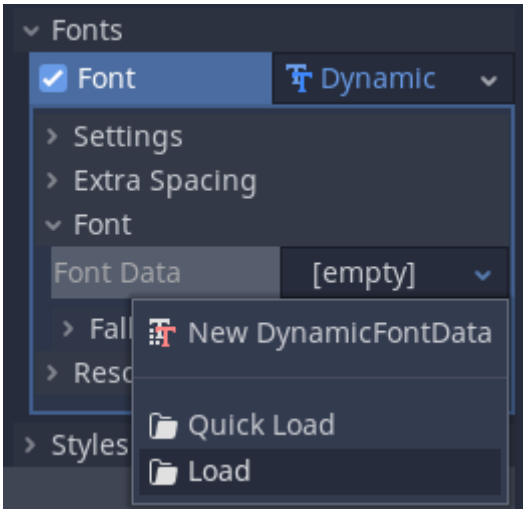
- Label с именем `ScoreLabel`.
- Label с именем `Message`.
- Кнопка с названием `StartButton`.
- Timer с именем `MessageTimer`.

Нажмите на `ScoreLabel` и введите число в поле `Text` в Инспекторе. Стандартный шрифт для узлов `Control` мал и плохо масштабируется. В ресурсы игры включен файл шрифта под названием "Xolonium-Regular.ttf". Чтобы использовать этот шрифт, сделайте следующее:

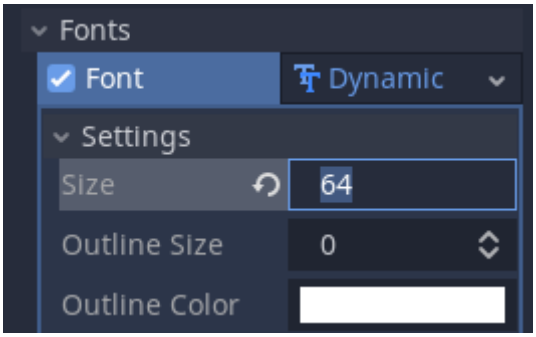
1. В разделе **«Переопределение темы»> «Шрифты»** щелкните пустое поле и выберите «Новый динамический шрифт».



2. Щелкните на "DynamicFont", который вы добавили, и в разделе **Font > FontData** выберите "Load" и выберите файл "Xolonium-Regular.ttf".



Установите свойство «Размер» в разделе `Settings`, `64` работает хорошо.

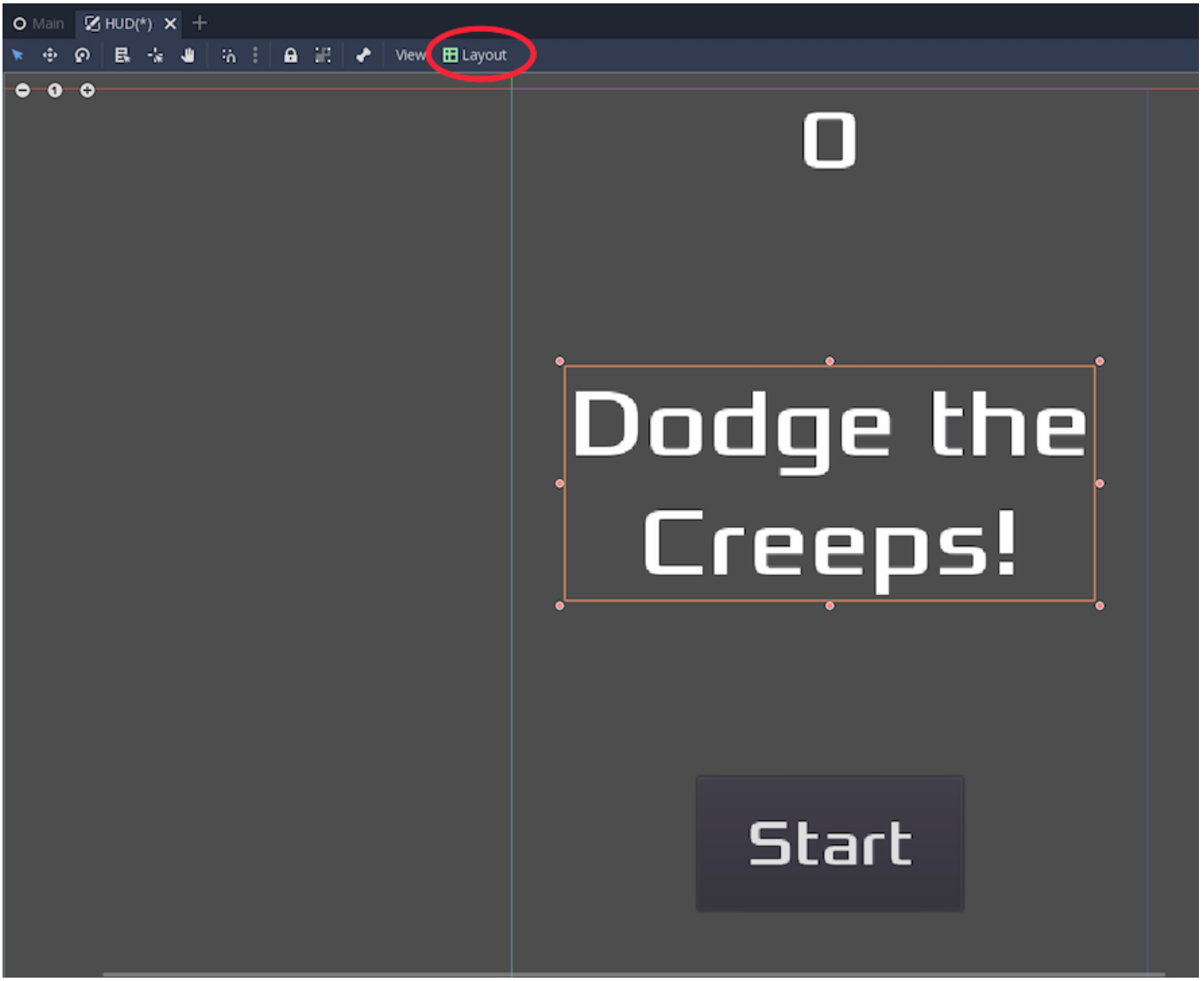


После того, как вы сделали это на `ScoreLabel`, вы можете щелкнуть стрелку вниз рядом со свойством «Шрифт» и выбрать «Копировать», а затем «Вставить» в то же место на двух других узлах управления.

Примечание

Якоря и поля: `Control` у узлов есть положение и размер, но у них также есть якоря и поля. Якоря определяют начало координат — опорную точку для ребер узла. Поля обновляются автоматически при перемещении или изменении размера управляющего узла. Они представляют собой расстояние от краев управляющего узла до его привязки.

Организуйте узлы, как показано ниже. Нажмите кнопку "Макет", чтобы задать макет для узла Control:



Вы можете перетаскивать узлы, размещая их вручную, либо используйте следующие параметры для более точного размещения:

ScoreLabel

- *Макет* : "Сверху по всей ширине"
- *Текст* : `0`
- *Выровнять* : "По центру"

Сообщение

- *Макет*: "По центру по всей ширине"
- *Текст* : `Уворачивайся от крипов!`
- *Выровнять* : "По центру"

- *Автообмотка* : "Вкл"

Кнопка Пуск

- *Текст* : `Старт`
- *Макет*: "Внизу посередине"
- *Маржа* :
  - Вершина: `-200`
  - Нижний: `-100`

В `MessageTimer` установите параметр `Wait Time` на `2`, а параметр `One Shot` на значение "Вкл".

Теперь добавьте этот скрипт в `HUD`:

GDScript C# C++

```
extends CanvasLayer

signal start_game
```

Сигнал `start_game` сообщает узлу `Main`, что кнопка была нажата.

GDScript C# C++

```
func show_message(text):
    $Message.text = text
    $Message.show()
    $MessageTimer.start()
```

Эта функция вызывается, когда мы хотим временно отобразить сообщение, такое как "Приготовьтесь".

GDScript C# C++

```
func show_game_over():
    show_message("Game Over")
    # Wait until the MessageTimer has counted down.
    yield($MessageTimer, "timeout")

    $Message.text = "Dodge the\nCreeps!"
    $Message.show()
    # Make a one-shot timer and wait for it to finish.
    yield(get_tree().create_timer(1), "timeout")
    $StartButton.show()
```

Эта функция вызывается, когда игрок проигрывает. Она покажет надпись "Game Over" на 2 секунды, затем произойдет возврат к основному экрану, и после короткой паузы появится кнопка "Start".

Примечание

Если вам нужно сделать паузу на короткое время, то альтернативой использованию узла Timer является использование функции SceneTree `create_timer()`. Может быть очень полезно добавлять задержки наподобие таких, как в вышеприведенном коде, где нам хотелось бы подождать немного времени, прежде чем показывать кнопку "Start".

GDScript C# C++

```
func update_score(score):
    $ScoreLabel.text = str(score)
```

Эта функция вызывается из `Main` каждый раз, когда изменяется количество очков.

Присоедините сигнал `timeout()` из `MessageTimer` и сигнал `pressed()` из `StartButton` и добавьте следующий код к новым функциям:

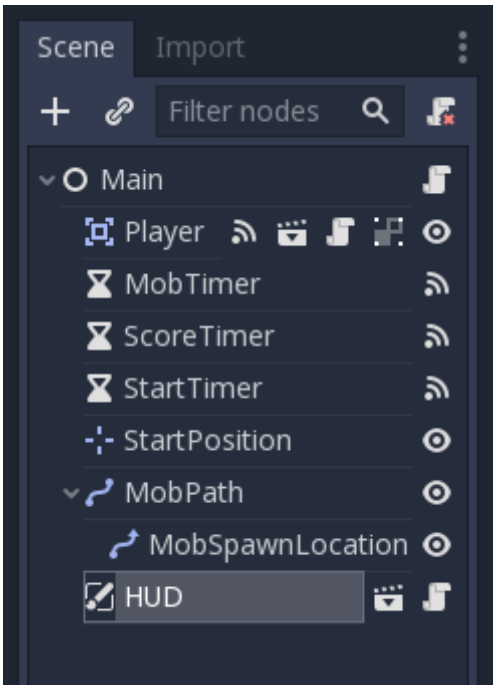
GDScript C# C++

```
func _on_StartButton_pressed():
    $StartButton.hide()
    emit_signal("start_game")

func _on_MessageTimer_timeout():
    $Message.hide()
```

Подключение HUD к Main

Теперь, когда мы закончили создание сцены `HUD`, вернитесь к `Main`. Инстанцируйте сцену `HUD` в `Main` подобно тому, как вы это делали со сценой `Player`. Дерево сцены должно выглядеть так, поэтому убедитесь, что вы ничего не упустили:



Теперь нам нужно подключить функционал `HUD` в наш `Main`-скрипт. Для этого потребуются некоторые дополнения к сцене `Main`:

Во вкладке "Узел" присоедините сигнал HUD `start_game` к функции узла Main `new_game()`, введя "new_game" в поле "Метод-приёмник" в окне "Подключить сигнал к методу" . Убедитесь, что в скрипте рядом с функцией `func new_game()` теперь появилась зелёная иконка подключения.

В `new_game()` обновим отображение счёта и выведем сообщение "Get Ready":

GDScript C# C++

```
$HUD.update_score(score)
$HUD.show_message("Get Ready")
```

В `game_over()` нам нужно вызвать соответствующую функцию `HUD`:

GDScript C# C++

```
$HUD.show_game_over()
```

Наконец добавьте это в `_on_ScoreTimer_timeout()`, чтобы синхронизировать отображение с изменением количества очков:

GDScript C# C++

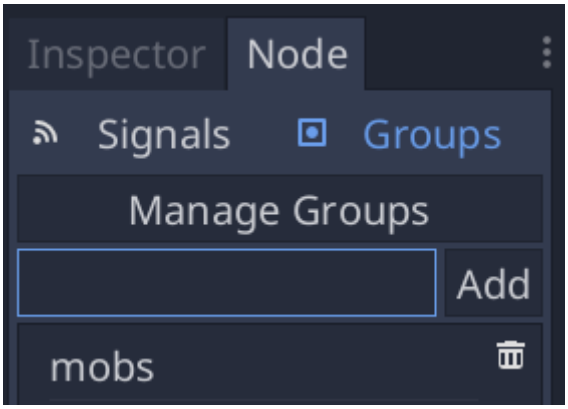
```
$HUD.update_score(score)
```

Теперь вы готовы к игре! Нажмите на кнопку "Запустить проект". Вам будет предложено выбрать основную сцену - выбирайте `Main.tscn`.

Удаляем старых крипов

Если вы играете до "Game Over", а затем сразу начинаете новую игру, то крипы из предыдущей игры могут все еще оставаться на экране. Было бы лучше, если бы все они исчезали в начале новой игры. Нам просто нужен способ сказать *всем* мобам, чтобы они удалились. Мы можем сделать это с помощью функции "group" ("группа").

В сцене `Mob` выберите корневой узел и нажмите вкладку "Узел" рядом с Инспектором (там же, где вы находите сигналы узла). Рядом с "Сигналы" нажмите "Группы", введите новое имя группы и нажмите "Добавить".



Теперь все мобы будут в группе "mobs". Затем мы можем добавить следующие строки к функции `new_game()` в `Main`:

GDScript C# C++

```
get_tree().call_group("mobs", "queue_free")
```

Функция `call_group()` вызывает каждую именованную функцию на каждом узле в группе - в этом случае мы говорим каждому мобу удалять себя.

Игра на данный момент по большей части готова. В следующей и последней части мы немного отполируем её с помощью добавления фона, проигрывания музыки и нескольких сочетаний клавиш.