Desenvolvimento com Motores de Jogos I

Aula 09 - Desenvolvendo a Primeira Fase - Parte 1

Apresentação

Olá, pessoal! Voltamos aqui, direto do nosso material, para mais uma aula superinteressante sobre o desenvolvimento de jogos 2D! Dessa vez, abordaremos um assunto importante, que vocês verão com mais detalhes em uma disciplina do módulo avançado: estudaremos Level Design, um assunto de Design de Jogos Digitais!

Ao longo das primeiras aulas, criamos um personagem bem completo e interativo, contendo animações, controles, pulo, física, etc. Agora, utilizaremos o nosso amigo robô para construir, de fato, o nosso jogo de plataforma. Para isso, modificaremos um pouco o nosso cenário, alteraremos o tamanho da cena como um todo e, por fim, adicionaremos... rufem os tambores... plataformas!

Alguns conceitos interessantes surgirão, também, junto a todas essas modificações que faremos. Conheceremos os Prefabs - elementos do Unity capazes de armazenar um objeto e suas propriedades para reutilização.

Há muita coisa legal para vermos nesse assunto e, por isso, mais uma vez, dividiremos o conteúdo em duas aulas. Nesta, estudaremos um pouco do processo de Level Design e os Prefabs. Na próxima, veremos toda a parte de código relacionada à criação de nosso cenário.

Mas deixa isso para a próxima aula! Nesta já tem muito assunto legal! Chega de introdução e vamos lá!

Objetivos

Ao final desta aula, você deverá ser capaz de:

- Compreender melhor o processo de Level Design;
- Começar a construir uma fase de plataforma no Unity;
- Criar e editar Prefabs para reutilizar elementos em uma cena.

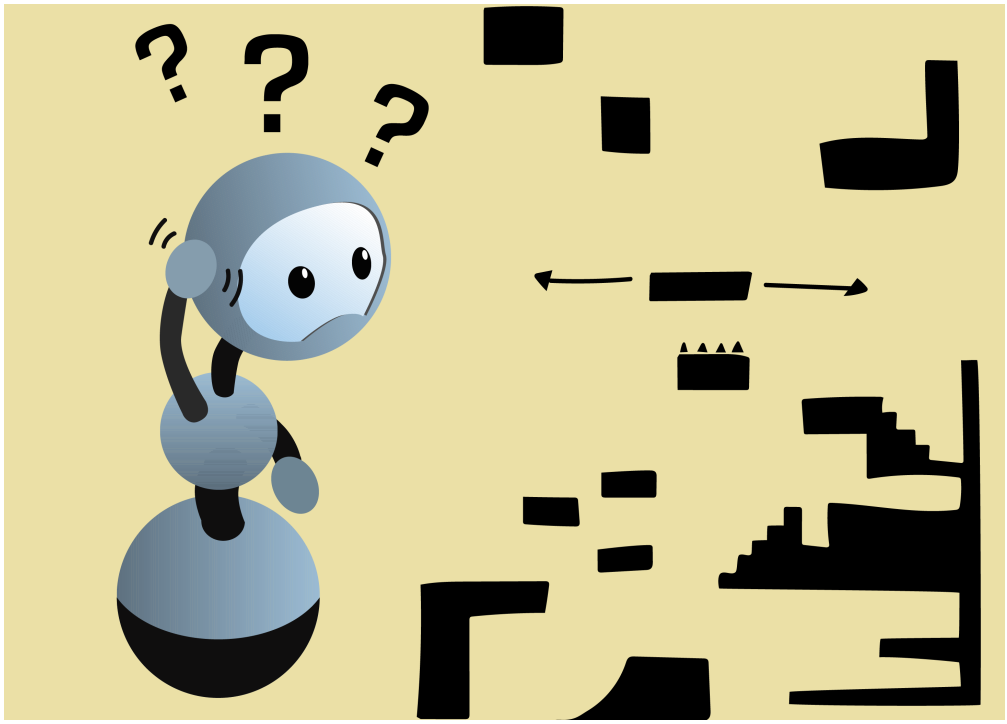
1. Level Design

Um dos aspectos mais importantes ao construirmos um jogo novo é preparar o Level Design. Essa etapa consiste na criação das fases que farão parte do seu jogo e no modo como elas devem ser encaradas pelos usuários.

A criação de uma fase é um processo complexo, o qual deve ser planejado e desenvolvido com cuidado, sempre visando o balanceamento e, principalmente, garantindo ao usuário ter uma boa experiência ao final de tudo. Não há nada mais frustrante do que você chegar até um ponto de um jogo e simplesmente não ter como sair de lá! A não ser que o jogador tenha feito besteira, não é? Aí ele merece! Vai lá reiniciar a fase, amigo! :P

Como todas as etapas no desenvolvimento de jogos, a criação de novos níveis passa por diversas fases e cada uma delas requer, para ser bem-sucedida, uma habilidade diferente. Passaremos aqui por cada uma dessas etapas, de modo que, ao fim das próximas aulas, tenhamos o nosso primeiro nível do jogo não só completamente pensado, como também implementado.

Mas calma! Temos um trabalho a ser feito antes mesmo de abrirmos o Unity, juntarmos os nossos recursos e começarmos a desenvolver! Precisamos, antes de tudo, saber o que desenvolveremos, concorda? Não faz muito sentido criarmos todos os tipos de plataforma existentes, ou começar a colocar elementos em nossa cena, e ver as consequências disso. Usualmente, perderemos mais tempo fazendo assim do que se tivéssemos antes pensado em um protótipo.



Fonte: Autoria própria.

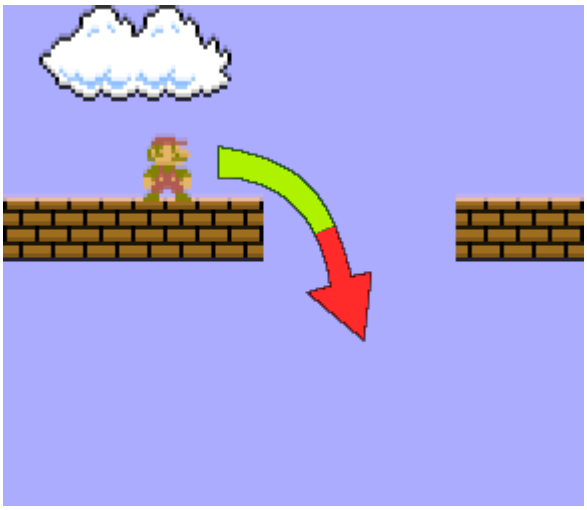
1.1 Pensando no Formato do Nível

Já sabemos das capacidades do nosso robozinho: ele anda para um lado e para o outro, pula e possui, também, um pulo duplo que o permite atingir distâncias verticais maiores. É importante saber disso, antes de agir, para garantir utilizarmos a maior parte das capacidades do personagem em nossas fases! Desse modo, podemos pensar melhor em uma coisa bastante importante para um jogo ter sentido: os desafios que incluiremos nele.

Como surgem as ideias para esses desafios? Bom... De acordo com o que veremos na disciplina de Design de Jogos, parte do trabalho de um bom designer é jogar! Quando jogamos, aprendemos diversas coisas sobre os jogos digitais. Isso serve para podermos nos inspirar nesses jogos e termos boas ideias de como faremos os nossos.

Baseado nisso, então, utilizaremos, para a nossa primeira fase, alguns desafios já conhecidos de outros jogos. O primeiro e mais clássico deles é ter, no chão, um buraco exigindo que o jogador precise pular para prosseguir no nível. É importante o buraco ter um tamanho que permita ao jogador pular com sucesso! E, também, não ser necessário tanto esforço, pelo menos no início, para conseguir pular a distância correta. Todo o espaço verde, na **Figura 1**, representa a área válida para pulo.

Figura 01 - Área de pulo sobre um buraco para o personagem Mario, da Nintendo, representada por uma seta.

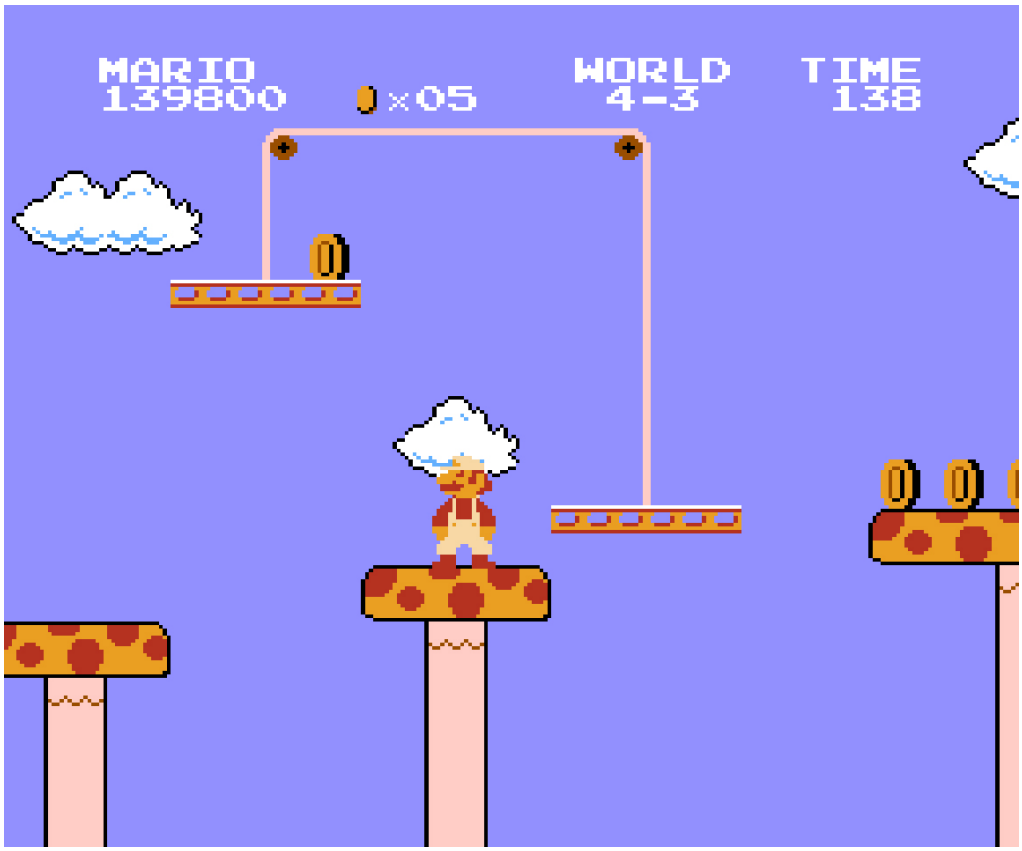


Fonte: Disponível em: http://devmag.org.za/2011/01/18/11-tips-for-making-a-fun-platformer. Acesso em: 05 de Mar. de 2017

Outro desafio que sabemos ser possível incluir no nosso jogo são as plataformas. Elas dificultam o caminho do nosso jogador e o fazem necessitar de alguma habilidade com o controlador para conseguir os pulos aos locais adequados. Assim como explicamos acerca dos buracos, é importante garantir que todas as plataformas posicionadas permitirão ao usuário alcançá-las.

As plataformas podem, também, possuir variações de comportamento para desafiar um pouco mais o usuário. Podemos ter plataformas que se movem, plataformas que giram, plataformas que caem, etc. O quanto a criatividade apurar e o Unity permitir, somos capazes de inventar!

Figura 02 - Plataformas móveis e estáticas no jogo Super Mario Bros. by Nintendo



Fonte: Super Mario Bros. by Nintendo

Ok. Sabemos tudo o que o nosso robô pode fazer e os tipos de desafios que podemos adicionar em nosso nível para o jogador poder se sentir desafiado ao jogar. O próximo passo, então, é criar o nosso primeiro nível! E criá-lo a partir desses desafios!

1.2 Prototipando o Nível

Existem algumas ferramentas que nos permitem criar níveis a partir de modelos predefinidos e de objetos adicionados à cena, posicionar tais objetos de acordo com o nosso gosto, fazer anotações... Além disso, há o lápis e o papel, que fazem algo similar muito bem! Vejamos, na **Figura 3**, a imagem do nosso primeiro nível, baseado nos desafios citados anteriormente!

Figura 03 - Protótipo manual do primeiro nível a ser desenvolvido em nosso projeto.

Fonte: Autoria própria.

Nossa! Que coisa linda! Não? Ok. :(

Esse protótipo manual serve, pelo menos, para você perceber que não precisa ser nenhum artista para prototipar um nível! O desenho feito por você ficará basicamente contigo e nunca será divulgado (a não ser que o coloque numa aula online). Servirá apenas como uma base para poder desenvolver o seu nível mais rapidamente, depois de já ter pensado em cada um dos desafios envolvidos.

E, por falar nisso, quais os desafios que foram escolhidos nesse protótipo? Já dá para ter uma ideia só olhando esse desenho?

O personagem iniciará sua jornada no canto esquerdo e buscará chegar até o objetivo final, isto é, o IMD, no canto direito. Como um bom jogo de plataforma mais antigo, o nosso jogo não terá qualquer tutorial. Devido a isso, é interessante, logo no primeiro nível, o usuário ser apresentado tranquilamente às mecânicas de jogo, até poder, com alguma facilidade, chegar ao final do nível.

Para isso, logo na primeira parte, o usuário encontrará uma pequena caixa, a qual terá de pular por cima, já utilizando a mecânica principal do jogo. Em seguida, o usuário terá uma outra caixa para pular, mas, dessa vez, terá também uma plataforma, após essa caixa, para perceber que pode pular de cima da caixa para outros locais. Se ele não se dirigir à plataforma, encontrará uma grande parede impedindo-o de seguir. A fim de passar pela parede, o jogador precisará utilizar a plataforma!

Passando da parede, o primeiro perigo mortal (ou não, pois ainda não implementaremos vidas/danos/inimigos, mas em breve serão implementados!): um buraco! O jogador, que já conhece a mecânica de pulo pelas caixas, deverá saltar o buraco e chegar ao outro lado para seguir. Nessa próxima parte, ele encontrará duas caixas para saltar novamente e, em seguida, outro buraco, um pouco maior. Nesse momento, o jogador necessitará de seu primeiro pulo duplo, a fim de chegar ao outro lado. Perceba que, como falamos inicialmente, esses pulos são bem simples, buscando apenas introduzir os jogadores às mecânicas. Nas outras fases do jogo, as mecânicas poderiam se complicar mais, com distâncias maiores e até mesmo saltos nas paredes, como você, desenvolvedor, desejar!

Sobrevivendo a isso, o jogador encontrará o seu primeiro puzzle a ser resolvido. Novamente, algo simples, mas mostrando a ele que às vezes é necessário recuar um pouco no cenário para conseguir avançar. Uma caixa dá acesso a uma plataforma que está atrás dela, permitindo um acesso a uma segunda plataforma, a qual está mais em cima a fim de, através desta, o jogador poder saltar na próxima parede. Daí em diante, todas as mecânicas iniciais já foram apresentadas e as coisas podem complicar um pouquinho mais.

A partir da segunda parede, o jogador deve executar um salto duplo sobre um buraco com uma parede do outro lado. Esse provavelmente será o primeiro ponto de derrota de vários jogadores que estão jogando pela primeira vez. Tudo bem! Lição aprendida para a próxima!

Quem conseguir saltar a parede executando o pulo duplo chegará até mais um pequeno espaço que o levará a um novo elemento do cenário: a plataforma movediça. Essa parte tem um buraco grande demais para ser saltado, mesmo com o pulo duplo, e o jogador deverá, com um pouco de paciência, esperar a plataforma móvel chegar perto dele a fim de poder pegar uma carona nela! Ele descobrirá, entretanto, no meio da jornada, que precisará se dirigir a uma segunda plataforma para conseguir completar o caminho todo, pois cada plataforma só faz metade do caminho. Mais um pequeno desafio!

Passando dessa etapa, chegamos ao final da fase... e aí complicamos mais um pouco! O usuário encontrará, agora, três plataformas, em sequência, como se formassem uma escada. Essas plataformas, no entanto, ao serem tocadas, simplesmente caem! E se o usuário não agir rápido, irá para o buraco próximo a elas. Mais uma chance perdida (em breve)!

Ao concluir esses pulos acelerados pelas plataformas que caem, o usuário deverá efetuar um último pulo longo, sobre uma última parede, para poder chegar ao fim do nível! E aí ele atingirá seu objetivo: chegar ao IMD! Uma alegria, não? :D

Com isso tudo, já temos o nosso primeiro nível pensado. Sabemos os desafios a serem postados aos usuários e o nível em que apresentaremos esses desafios. Sabemos, também, quais componentes necessitaremos desenvolver para isso

funcionar e como configuraremos o comportamento de cada um desses elementos para cumprirem seus objetivos nos níveis. Assim, podemos, finalmente, pôr a mão na massa!

Abaixo, um resumo dos componentes que precisaremos trabalhar nesse protótipo:

- Backgrounds
- Chãos espaçados com buracos
- Plataformas
    - Caixa no chão
    - Plataforma fixa
    - Paredes
    - Plataformas móveis
    - Plataforma que cai
- Começo e fim da fase

Além disso, faremos algumas alterações no nosso robô para ele poder se comportar melhor de acordo com o cenário desenvolvido. Todas essas modificações criarão a nossa primeira fase completa! Temos bastante trabalho, portanto, a fim de facilitar a compreensão de todos, dividiremos o desenvolvimento em duas aulas!

Perceba, também, que demos uma cor para cada um dos tipos de plataforma do cenário e duas cores para as plataformas móveis. Fizemos isso objetivando facilitar o entendimento, em conformidade com os modelos a serem utilizados para desenvolver nosso primeiro nível. Criaremos tudo a partir de um sprite vazio e utilizaremos essa chave de cores a fim de representar cada um dos elementos apresentados!

2. Criando o Level no Unity

Antes de tudo disponibilizamos aqui, novamente, o que desenvolvemos até a aula passada, para todos poderem partir de um mesmo ponto, caso não estejam acompanhando progressivamente. Baixem e adicionem a um novo projeto, sem esquecer de alterar as tags e layers, se algo não funcionar!

Ao abrimos o nosso projeto como deixamos ao fim da aula passada, encontraremos o nosso robô no cenário composto pelo background e pelo chão. Começaremos as melhorias a partir desse ponto!

2.1 Criando o Background

O primeiro passo será criar um background capaz de cobrir toda a fase. Uma técnica muito comum utilizada no design de níveis ao se ter apenas um background, no entanto, é repeti-lo ao longo da fase, quantas vezes forem necessárias.

Perceba que estamos desenvolvendo um jogo de plataforma, logo, as fases a serem desenvolvidas terão uma formatação definida, com um tamanho definido e plataformas cuidadosamente posicionadas. Não se trata de um jogo de corrida infinita ou de um cenário gerado proceduralmente. Por essas razões, a utilização da técnica citada se encaixa muito bem. Podemos simplesmente duplicar o background quantas vezes forem necessárias e posicioná-lo de maneira a criar um plano adequado no qual o nosso jogo será passado.

Também é importante não deixar nenhuma parte do cenário descoberta, nem mesmo deixar qualquer espaço entre os backgrounds. Além disso, evitar falhas de continuidade é imprescindível, uma vez que elas fazem o usuário perceber uma quebra brusca, e isso pode atrapalhar a experiência.

Por sorte, o background que estamos utilizando pode ser repetido com qualidade, se posicionarmos cada peça adequadamente, encaixando os prédios de fundo uns aos outros. Vamos cuidar dessa primeira parte, então!

Alguns desenvolvedores preferem utilizar um elemento vazio, o qual é apenas responsável por guardar repetições de um mesmo elemento, mantendo, assim, a cena no Unity mais organizada e facilitando o entendimento do que está

acontecendo naquele ambiente. Também tenho esse costume e gostaria que vocês tivessem!

O problema dessa abordagem, no entanto, parte de um aspecto visto anteriormente, em nossa aula sobre a câmera. Ao colocar um objeto como filho de outro, o espaço em que ele passa a ser posicionado é o espaço local do filho, e não mais as coordenadas globais. Por isso, precisamos tomar cuidado com as posições escolhidas! Nesse caso, o posicionamento visual sempre ajuda. Basta clicar no objeto, selecionar a opção de mover e colocá-lo exatamente onde gostaríamos de o ver.

Para começar a criação de nosso level, adicionaremos à nossa cena um elemento vazio através do menu GameObject -> Create Empty. Em seguida, posicionaremos esse objeto vazio na origem, clicando nele e colocando suas coordenadas para (0,0,0). Feito isso, o próximo passo é renomear o objeto para Backgrounds. É nesse objeto vazio que guardaremos todas as repetições de nosso background, posicionando-as de acordo com suas coordenadas. Como inicialmente colocamos o objeto na origem do mundo, as coordenadas do background se manterão as mesmas!

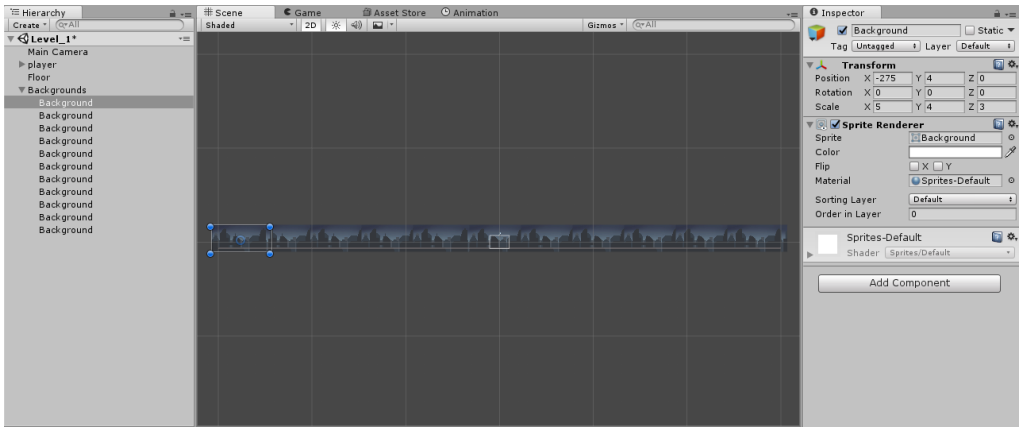
O passo seguinte, agora, é criar backgrounds o bastante para conseguirmos cobrir toda a nossa cena. Quantos serão necessários para isso? Não saberíamos inicialmente! Poderíamos adicioná-los à medida que precisássemos. No entanto, como já fiz o projeto completo, sei que serão necessários onze backgrounds para podermos fazer todo o nível. Assim, criaremos os onze e os posicionaremos adequadamente. A maneira mais fácil de fazer isso é configurar um background como queremos e simplesmente duplicá-lo o quanto for necessário!

Após alguns testes, chegamos à conclusão de que a melhor configuração para o background é utilizar a escala (5,4,1), com a posição sendo (X, 4, 0), na qual X corresponde ao posicionamento em X de cada um deles. Como faremos o nosso nível de -300 a 300, os valores de X variarão dentro dessa faixa. Clique no background e o arraste para o objeto vazio, tornando-o, assim, filho dele.

Feito isso, configuraremos o nosso background atual para o transform (-275, 4, 0), com a escala citada. Em seguida, podemos clicar com o botão direito no GameObject e selecionar a opção Duplicate. Desse modo, uma nova instância do

Background aparecerá, exatamente igual a anterior. Repetiremos esse processo mais nove vezes, totalizando onze backgrounds. Todos eles terão o mesmo tamanho já definido e a mesma posição. Precisaremos, apenas, variar o X para criar o nosso cenário! Utilizaremos os seguintes valores para cada um dos onze X: -275 | -225 | -170 | -115 | -60 | 0 | 55 | 110 | 165 | 220 | 275. O resultado desse posicionamento pode ser visto na **Figura 4**.

Figura 04 - Backgrounds posicionados, filhos de um objeto vazio.



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 24 de fev de 2017.

Alteraremos, também, a posição do chão para (0,-6.5,0) e sua escala para (600, 1, 1). Com isso, nosso personagem já pode percorrer todo o cenário no chão gigante! Embora ele ainda não ter nada para fazer, podemos admirar a paisagem e utilizar o espaço maior a fim de ver a animação de caminhada várias vezes! :D

Os que resolverem se aventurar pela longa caminhada, chegarão até o início da fase (ou até o fim, dependendo do lado escolhido) e aí... nada! Caiu! Isso não costuma acontecer em jogos por aí, não é? Eles, usualmente, possuem um começo e um fim na fase, de modo a não ser possível passar de determinados pontos. Adicionaremos, agora, um começo e um fim ao nosso jogo!

2.2 Adicionando um Começo e um Fim à fase

Já temos uma fase construída! Tudo bem ela não ser a mais divertida do mundo, permitindo a você apenas andar para um lado e para o outro, mas já funciona! Falta, no entanto, haver um fim e um começo nessa fase. Pontos em que não poderemos mais voltar ou avançar. O que fazer com esses pontos veremos nas aulas seguintes, agora, porém, aprenderemos como podemos definir cada um deles.

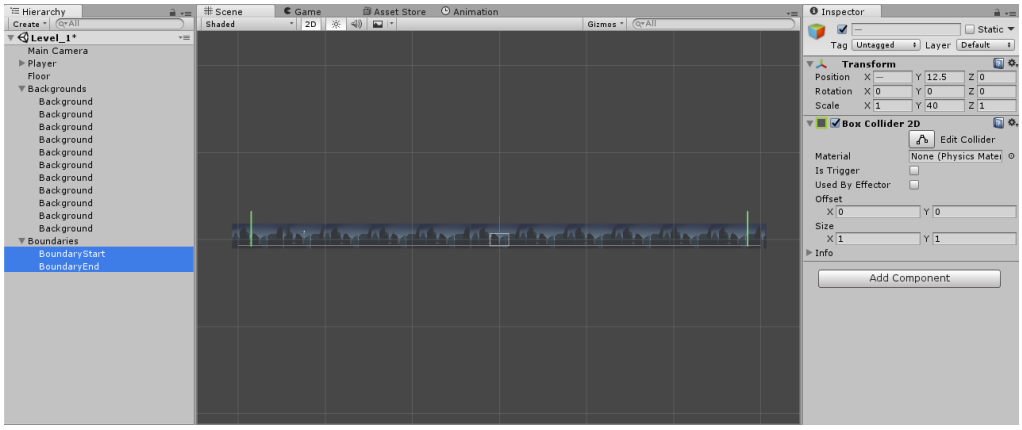
As bordas, ou limites, são conhecidas, em inglês, como Boundaries. Devemos adicionar uma dessas, pelo menos, ao início e ao fim da fase. Elas funcionam como pontos invisíveis de bloqueio ao usuário, indicando não haver nada além dali. Podemos, também, utilizar esses pontos como marcadores, mas não é o objetivo desta aula. Por enquanto, apenas bloquearemos o personagem de se mexer.

Para isso, como fizemos com os backgrounds e com todos os objetos adicionados nesta aula, iniciaremos criando um novo elemento vazio, posicionando-o em (0,0,0) e alterando o seu nome, dessa vez, para Boundaries. Em seguida, criaremos mais dois objetos vazios, agora como filhos de Boundaries, para podermos definir os limites da fase. Devemos chamar esses objetos filhos de BoundaryStart e BoundaryEnd.

A esses elementos filhos, adicionaremos um Box Collider 2D em cada, a fim de o player colidir com eles ao alcançá-los e não poder passar dali. Esses objetos devem ser altos, para evitar que o personagem consiga saltar por cima. Eles não precisam, no entanto, serem largos.

Para cumprir esses requisitos, configuraremos os objetos com a escala (1,40,1) e com o posicionamento, para o BoundaryStart, (-250, 12.5, 0) e, para o BoundaryEnd, (285, 12.5, 0). Assim, o personagem será limitado ao ponto -250 no começo e ao ponto 285 no final. Chegando em qualquer uma dessas localidades, ele colidirá com o nosso objeto invisível e por lá ficará. A **Figura 5** mostra, marcados em verde devido aos seus colisores, a adição desse grupo em nossa hierarquia e o posicionamento em que os elementos estão em nossa cena.

Figura 05 - Elementos de fronteira posicionados na cena para evitar que o personagem saia da área de jogo desenvolvida.



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt/. Acesso em: 24 de fev de 2017.

Com a adição desses elementos, nosso personagem já está confinado à nossa fase. Mesmo sendo apenas um nível vazio, com um background e um chão bem longo, já é possível garantir que o nosso personagem não sairá de lá... vivo! Muahahaha

O próximo passo é exatamente este: daremos ao personagem uma maneira para sair de lá. Abriremos buracos no chão para ele poder cair. O que acontecerá depois de sua queda são cenas dos próximos capítulos, mas os buracos a serem adicionados veremos agora!

2.3 Reposicionando o Chão para Criar os Buracos

Agora que nosso cenário já tem uma cara de jogo, podemos começar de fato a adicionar os desafios. Como vimos na **Figura 3**, o nosso esboço indica que teremos um total de seis pedaços de chão em nossa fase. Esses pedaços surgem ao separarmos o chão em duas partes a fim de haver um buraco entre elas, criando, assim, a necessidade de o jogador pular.

Faremos com o chão o mesmo que fizemos com os dois componentes anteriores: criaremos um objeto vazio, o posicionaremos em (0,0,0) e adicionaremos o nosso antigo objeto *Floor*, o qual nos acompanha desde a segunda aula, como filho desse novo objeto vazio. Podemos chamar o objeto vazio de Floors. Posicionaremos o nosso *floor* em (X, -6.5, 0), com a escala de (X', 1, 1). No caso desses objetos, diferentemente do que vimos anteriormente, além da posição deles variarem, as escalas também variarão, pois eles possuem tamanhos diferentes, de acordo com o local onde o buraco será aberto.

O objeto *Floor* deve ser duplicado cinco vezes, totalizando seis objetos. Esses objetos devem ser colocados nas seguintes posições:

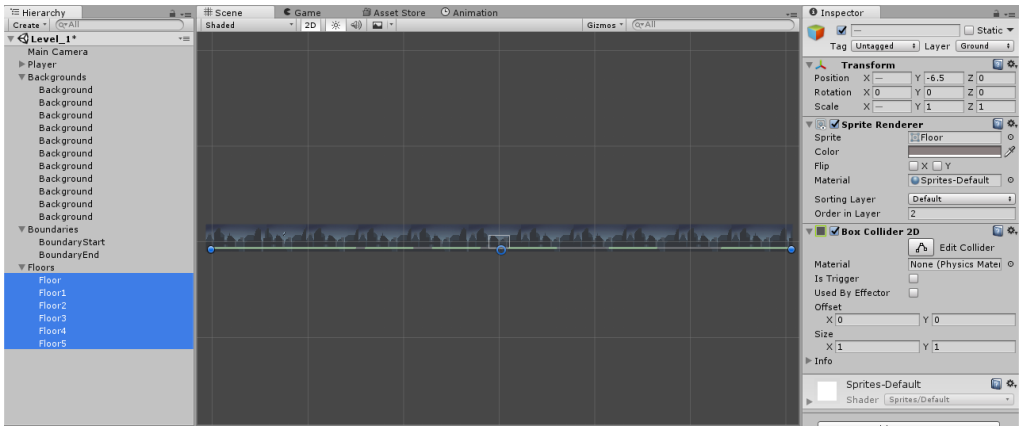
Objeto	Posição	Escala
Floor	(-200, -6.5, 0)	(100,1,1)

Objeto	Posição	Escala
Floor 1	(-115, -6.5, 0)	(50,1,1)
Floor 2	(-45, -6.5, 0)	(50,1,1)
Floor 3	(35, -6.5, 0)	(50,1,1)
Floor 4	(140, -6.5, 0)	(50,1,1)
Floor 5	(270, -6.5, 0)	(75,1,1)

Tabela 1 - Posicionamento dos objetos de chão.
Fonte: Autoria própria.

Com isso, chegamos ao que vemos na **Figura 6**. Percebam a Layer da qual os objetos todos fazem parte!

Figura 06 - Buracos adicionados através da criação de múltiplos objetos Floor.



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt. Acesso em: 24 de fev de 2017

Percebam como há um aumento gradativo do espaço entre os buracos. Esses valores foram selecionados justamente com esse intuito, baseado no que esboçamos na **Figura 3**. Sendo assim, é possível passar pelo primeiro buraco com um pulo simples, pelo segundo apenas com um pulo duplo. O terceiro, por sua vez, requer que você venha de um local mais alto. O penúltimo precisa de uma das

plataformas móveis, e o último precisa da escada que cai. Cada um deles é um pouco maior e acrescenta um pouco mais ao desafio! Também é possível, com essa configuração de chão, chegar aos dois lados da fase, de início e fim.

Figura 03 - Protótipo manual do primeiro nível a ser desenvolvido em nosso projeto.

Fonte: Autoria própria.

Teste um pouco o jogo agora e veja se consegue passar pelos buracos! Ah! E uma dica de *cheat code* que você desenvolvedor pode usar: caso esteja caindo em algum dos buracos, clique no botão de pause, ao lado do play, e altere o valor do Y do jogador. Isso fará o personagem subir novamente e você poderá continuar o seu teste sem precisar reiniciar tudo. Como a alteração é feita durante o play, ela não será refletida após encerrar o modo de jogo. Legal, não? :)

Atividade 01

1. Mova o jogador para a posição (-225, 10, 0) e tente completar a fase do modo que o jogo está atualmente. É possível?

2.4 Criando as Caixas do Jogo

No começo da aula, discutimos os componentes que adicionaríamos ao nosso jogo nas próximas aulas a fim de criarmos nosso primeiro nível. Já configuramos o background, o chão e os limites da fase. Começaremos, agora, a adicionar as plataformas necessárias para chegarmos ao protótipo.

As plataformas serão todas criadas com sprites simples, da mesma maneira como fizemos para criar o chão. A diferenciação entre os seis tipos de plataforma que teremos em nosso jogo acontecerá apenas através de suas cores, além, é claro,

de seu comportamento e apresentação.

A primeira plataforma que adicionaremos ao nosso jogo é a mais simples de todas - uma caixa no chão para o usuário poder subir ou passar sobre ela com apenas um pulo simples.

A importância desses objetos mais simples é grande, uma vez que, no início, podem servir como um local de aprendizado para o jogador em relação aos controles e, mais adiante, podem ser a base de puzzles mais complicados e passos mais desafiadores.

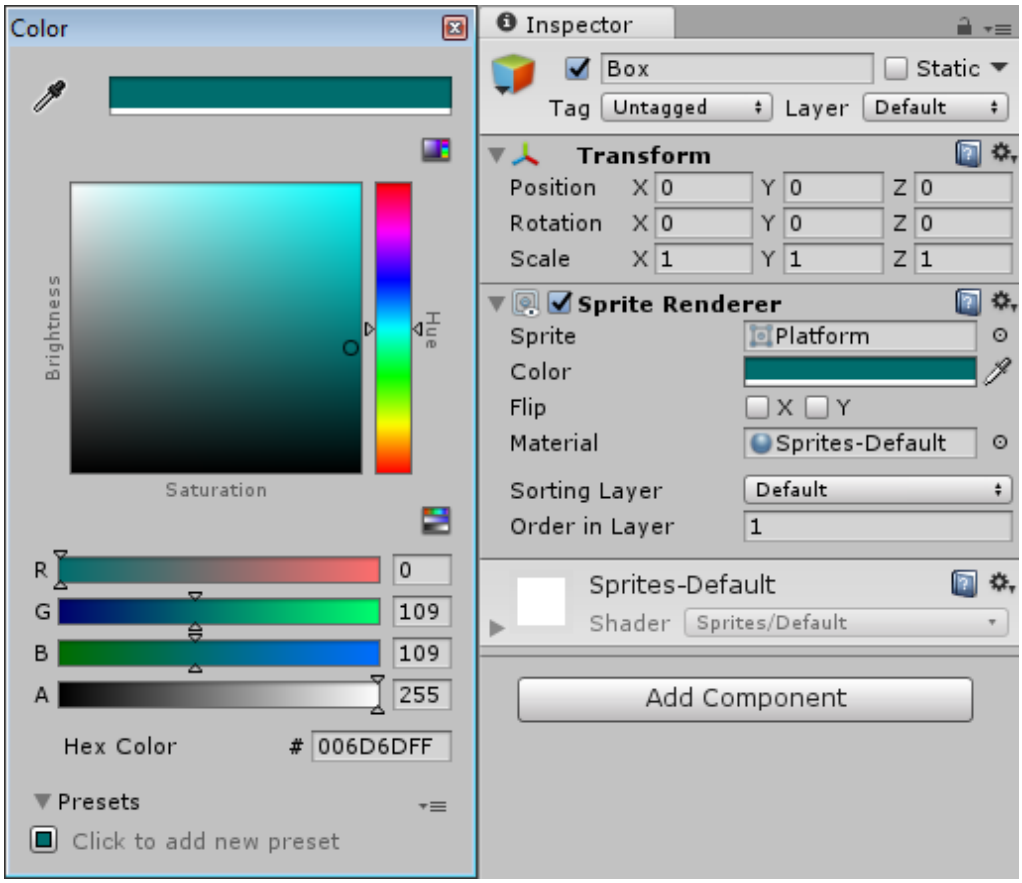
Novamente, iniciaremos como fizemos com os outros componentes. Criaremos um novo objeto vazio, chamado Platforms, e adicionaremos, dentro dele, todas as plataformas que utilizaremos no jogo. Não esqueça de posicionar o objeto em (0,0,0) para igualar as coordenadas de seus filhos com as coordenadas do mundo!

Em seguida, adicionaremos uma plataforma como filha desse objeto. No entanto, dessa vez utilizaremos, para a plataforma, uma abordagem um pouco diferente do que vínhamos utilizando anteriormente. Criaremos só uma plataforma e, a partir desta, criaremos um Prefab a fim de construir as outras. Primeiro, criaremos a plataforma, para depois discutirmos como funciona um Prefab e, então, criá-lo!

Em nossa pasta Assets -> Sprites criaremos o nosso novo sprite. Para fazer isso, clique com o botão direito na pasta Sprites e selecione a opção Create -> Sprite -> Square. Chame de Platform, o novo sprite criado. Clicaremos sobre ele e o arrastaremos até o nosso novo objeto vazio Platforms, na aba Hierarchy. Isso criará um novo sprite quadrado, com o nome Platform, em nossa cena. Altere o nome do GameObject criado para Box. A partir desse objeto será construída a nossa primeira caixa.

A primeira coisa a ser feita é alterar a cor para aquela que havíamos demonstrado em nossa lista, feita no início da aula. A cor da caixa, especificamente, era R=0, G=109, B=109 ou #006D6DFF em Hex. Para acessar a troca de cores através de RGB ou Hex, basta clicar na propriedade Color, do Sprite Renderer de nosso objeto, e então digitar esses valores no campo indicado pelas letras R, G e B ou no campo Hex Color. A **Figura 7** exibe essa operação.

Figura 07 - Escolhendo um valor para a propriedade Color.



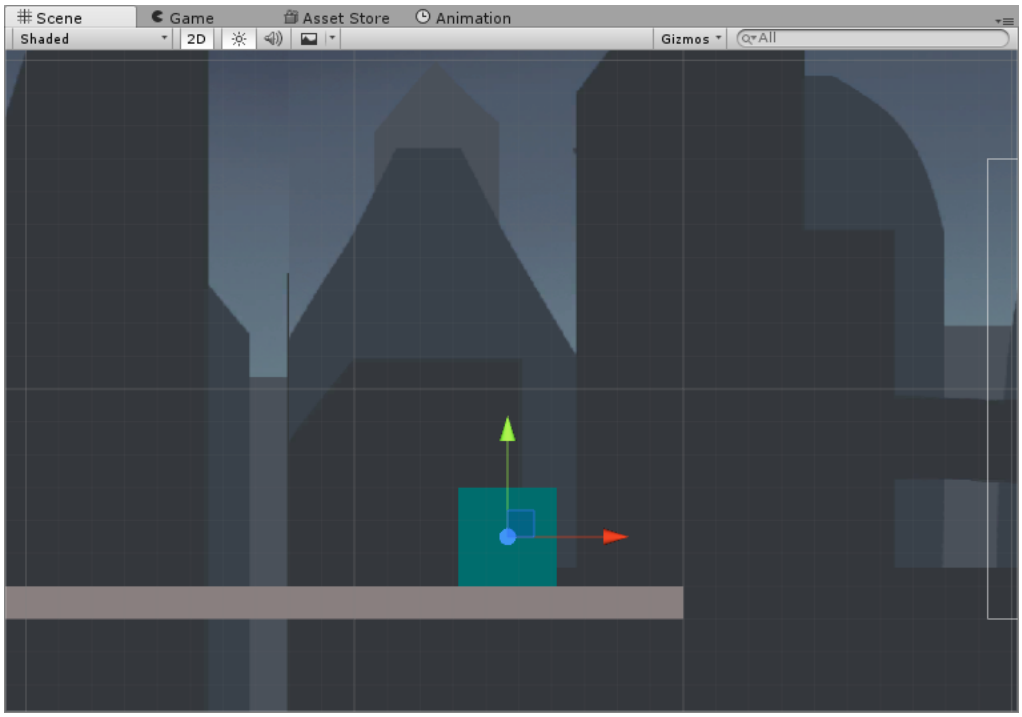
Fonte: Captura de tela do Unity. Disponível em:https://unity3d.com/pt. Acesso em: 24 de fev de 2017

Com a cor escolhida, o próximo passo é posicionar o nosso objeto adequadamente e o escalar. Para esse tipo de plataforma, utilizaremos a posição (X, -4.5, 0), com a escala (3,3,0). Essa escala permite ao jogador, se as configurações do Robô estiverem como estão no meu projeto, pular em cima da caixa utilizando apenas um pulo.

Já a posição -4.5 garante que a caixa parecerá em contato com o chão, sem necessariamente parecer dentro dele. Inclusive, para evitar qualquer problema de corte visual, a plataforma deve ser posicionada na segunda camada, através da propriedade Order in Layer, com o valor 1. O background faz parte da primeira camada, ficando sempre atrás de tudo e tendo o valor 0 dessa propriedade, enquanto o chão fica à frente e deve ter, em cada um dos objetos Floor, o valor 2 para a ordem.

Outro detalhe muito importante diz respeito à Layer! A camada da qual a caixa faz parte deve ser a mesma do chão, para permitir ao personagem pular a partir dela e permitir à animação reconhecer que o personagem está apoiado em um solo e poder ser exibida corretamente, concorda? Então, vamos lá na caixa, na opção Layer, alterar o seu valor para Ground, que é a camada representante do solo e buscada no script de nosso Player. Com isso, a caixa está criada e (quase) pronta para ser utilizada no jogo. Podemos vê-la posicionada na **Figura 8**.

Figura 08 - Caixa posicionada no cenário da primeira fase.



Fonte: Captura de tela do Unity. Disponível em:https://unity3d.com/pt. Acesso em: 24 de fev de 2017

Agora, no entanto, faremos algo um pouco diferente com essa caixa. Após configurar sua cor, sua ordem de renderização e suas propriedades de Transform, estamos prontos para transformá-la em um Prefab de objetos desse tipo, ao invés de sair duplicando-a. Mas o que é um Prefab?

Fonte: Autoria própria.

2.5 Prefabs

Muitas vezes, ao construir uma cena, você possui objetos repetidos nela. Podem ser objetos simples, como a caixa criada agora, ou até objetos mais complexos, como os scripts, elementos de IA, renders e tudo mais. Podemos, como fizemos com os Backgrounds e com os Floors, multiplicar esses objetos quando desejarmos, para criarmos cópias. Agora, imagine se você quisesse adicionar um novo elemento a esse objeto. Você precisaria modificar um por um, correto? Será que existe alguma maneira de modificar todos de uma vez e manter uma unidade entre eles, visto que são, basicamente, instâncias de um mesmo objeto? Bem. Existe sim! Qualquer um desses objetos pode ser transformado em um Prefab!

O manual oficial do Unity define um Prefab como "um Asset que permite a você guardar um objeto do tipo GameObject completo, com seus componentes e propriedades. Um Prefab age como um template a partir do qual você pode criar novas instâncias de um objeto em sua cena. Qualquer modificação que seja feita em um Asset do tipo Prefab é também, automaticamente, feita em todas as instâncias que foram criadas a partir daquele Asset (n.e. exceto modificações no Transform). É possível também sobrescrever os valores de cada instância individualmente".

Ou seja, um Prefab pode ser resumido como um GameObject que vira Asset! E isso é fantástico! Nós criamos a nossa caixa e, agora, podemos criar um Prefab a partir dela, bastando arrastarmos esse Prefab para a nossa cena, da mesma maneira como já fizemos várias vezes com Sprites, e então uma nova caixa será criada, igualzinha à caixa inicial que planejamos. E se eu tiver uma nova ideia? Quiser trocar a cor da caixa? Adicionar um novo script? Não há problema! É só atualizar o Asset Prefab criado a partir do GameObject e todos os objetos instanciados a partir

daquele Prefab se atualizarão também. Assim, você garante que todos os objetos manterão sempre o mesmo padrão e serão alterados da mesma maneira! Muito interessante, não?

E não é só isso! ~~Ligando agora você ainda leva inteiramente grátis...~~ A criação de Prefabs é supersimples e intuitiva! Quer ver? Criaremos agora um Prefab a partir da nossa caixinha a fim de todas as outras serem feitas com base nessa mesma instância do objeto, facilitando o controle de um modo geral.

Para começar, crie uma nova pasta dentro de nossa pasta Assets, na aba Project, chamada Prefabs. Acesse essa pasta e deixe a aba Project aberta. Agora clique na nossa Box, a qual está na Hierarchy, e arraste-a até a aba Project, dentro da pasta Prefabs criada. Pronto! Esse Assets que acaba de ser criado representa justamente a nossa caixa! Que fácil! Que útil!

E agora? Como podemos criar, em nosso cenário, as caixas que precisamos? Bem, olhando na **Figura 3**, lá atrás, temos um total de cinco caixas na fase completa. Já temos a caixa usada como referência para o Prefab, então, adicionaremos mais quatro. A fim de fazer isso, visto que já temos o objeto Platforms para reunir as plataformas, basta selecionarmos o Prefab criado e arrastá-lo até o objeto Platforms. Isso criará uma nova instância do Prefab como filho desse objeto e nós poderemos posicioná-lo de acordo com o que for necessário!

Figura 03 - Protótipo manual do primeiro nível a ser desenvolvido em nosso projeto.

Fonte: Autoria própria.

Adicione cinco filhos e altere suas posições em X para: -200 | -180 | -130 | -110 | -44. Todas as outras propriedades devem ser mantidas como estão, uma vez que já foram configuradas para o Prefab inicial. Excelente!

Teste agora o que fizemos! Dê Play no seu jogo e pule em cima de uma dessas caixas para celebrar a nossa vitória! YEAH!

Figura 09 - Robô entrando na caixinha.

Fonte: Captura de tela do Unity. Disponível em:https://unity3d.com/pt. Acesso em: 24 de fev de 2017

Ops...

Não deu certo, né? Passamos direto por dentro da caixa! É que faltou um detalhe muito importante, o qual alguns de vocês até perceberam antes mesmo de mostrarmos a **Figura 9**... Faltou colocar um colisor em nossa caixinha! E agora? Temos cinco caixas para modificar! Que trabalheira!

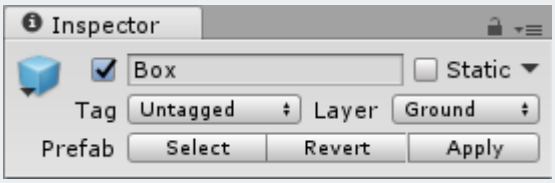
Não

É um Prefab, jovem! Basta modificarmos o próprio Asset, dentro de nossa pasta Prefabs, para todas as instâncias serem alteradas, correto? Então vamos fazer isso!

Selecione, na aba Projects, na pasta Prefabs, o Prefab da caixinha que acabamos de criar. Ao fazer isso, o Inspector mostrará as propriedades **do Prefab**, e não mais de uma instância. Assim, basta fazer aquilo que já estamos acostumados... Add Component -> Physics 2D -> Box Collider. Pronto! Agora clique em cada uma das caixinhas adicionadas em nosso jogo. Todas possuem, agora, Colliders, correto? Simples demais, não? Sempre, ao criarem um novo GameObject em suas fases, pensem: será necessário repetir esse objeto? Deveria criar um Prefab? A resposta a essas perguntas pode te economizar bastante tempo e trabalho! Utilize, quando for possível, essa grande facilidade disponibilizada pelo Unity.

Ah! E outra coisa interessante! Também é possível alterar o nosso Prefab a partir das instâncias, ao invés de alterarmos o Asset criado como Prefab. Para isso, basta selecionar uma das instâncias daquele Prefab, alterar os valores desejados e, então, apertar o botão Apply, como visto na **Figura 10**. Desse modo, todas as propriedades alteradas naquela instância, com exceção do Transform, são alteradas no Prefab e, consequentemente, em todas as outras instâncias criadas a partir daquele Prefab. Bem útil, também!

Figura 10 - Botões adicionais recebidos por uma instância do Prefab.

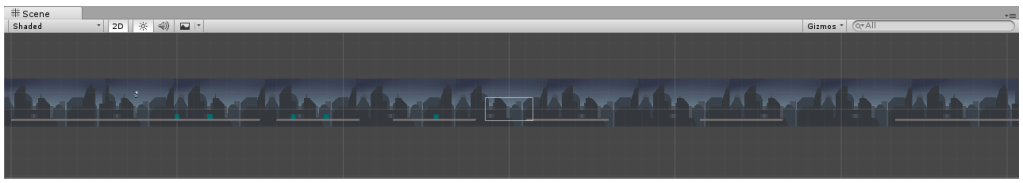


Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt. Acesso em: 24 de fev de 2017

Além disso, também é possível reverter alterações feitas em uma instância para os valores salvos em um Prefab. Isso é feito através do botão Revert, também visto na **Figura 10**.

Dessa maneira, temos parte de nosso cenário já definido. Encerramos esta aula por aqui, com o cenário parcialmente desenvolvido, como visto na **Figura 11**. Na próxima aula, posicionaremos as plataformas estáticas que faltam, já utilizando esse conceito de Prefabs, e, em seguida, codificaremos alguns scripts para criarmos as plataformas móveis! Falta, também, melhorar o nosso Player, pois o comportamento dele não está muito adequado dentro da fase. Há, ainda, muito assunto legal antes de encerrarmos essa parte de Level Design! Até lá, pessoal! o/

Figura 11 - Fase desenvolvida até o momento para o jogo.



Fonte: Captura de tela do Unity. Disponível em: https://unity3d.com/pt. Acesso em: 24 de fev de 2017

Leitura Complementar

Manual do Unity sobre Prefabs

<https://docs.unity3d.com/Manual/Prefabs.html>

11 Tips for Making A Fun Platformer

<http://devmag.org.za/2011/01/18/11-tips-for-making-a-fun-platformer/>

Resumo

Nesta aula, começamos o Level Design do nosso jogo de plataforma. Relembramos um pouco o conceito de Level Design e o modo como este é feito. Em seguida, começamos as modificações em nosso projeto, a fim de o jogo ganhar uma primeira fase com desafio, um começo e um fim. Criamos elementos vazios que funcionaram como agrupadores aos objetos de mesmo tipo e, então, criamos vários backgrounds, diversos pedaços de chão, fronteiras do mapa e, por fim, a primeira de nossas plataformas.

Para facilitar a criação de nossas plataformas, definimos um Prefab, a partir da primeira plataforma que criamos, para podermos simplesmente criar novas instâncias desse Prefab, ao invés de duplicar manualmente todos os objetos. Vimos, ainda, como é fácil alterar objetos que partem de um mesmo Prefab, alterando apenas o asset, e não cada uma das instâncias.

Também durante a aula, aprendemos o que é um Prefab e como os seus conceitos ajudam o desenvolvedor quando há a necessidade de replicar objetos em uma mesma cena.

Finalizamos a aula vendo um pouco como o nível tinha ficado (o projeto pode ser obtido aqui) e nos preparamos para a próxima aula, na qual adicionaremos as plataformas que faltam, incluindo as que possuem scripts, e também melhoraremos o controle do robô, para ele ficar mais fiel à realidade.

Autoavaliação

1. Qual a importância, para o desenvolvimento de uma nova fase, do Level Design e da criação de protótipos?
2. Qual o motivo de criarmos objetos vazios em nossa cena?
3. O que são Prefabs? Para que devem ser utilizados?
4. É possível reconfigurar um Prefab através de uma instância? Como fazer?

Referências

Documentação oficial do Unity. Disponível em: https://docs.unity3d.com/Manual/index.html

Tutoriais oficiais do Unity. Disponível em: https://unity3d.com/pt/learn/tutorials

RABIN, Steve. **Introdução ao Desenvolvimento de Games,** Vol 2. CENGAGE.