BlockClimber 2D – Documentação

Jogo desenvolvido para teste da **VirtualPlanet**

**12/05/2021 – Versão inicial lançada [v0.1.0]**

Versão inicial lançada para atender ao teste da VirtualPlanet, a versão é uma *alpha* que conta com as seguintes features:

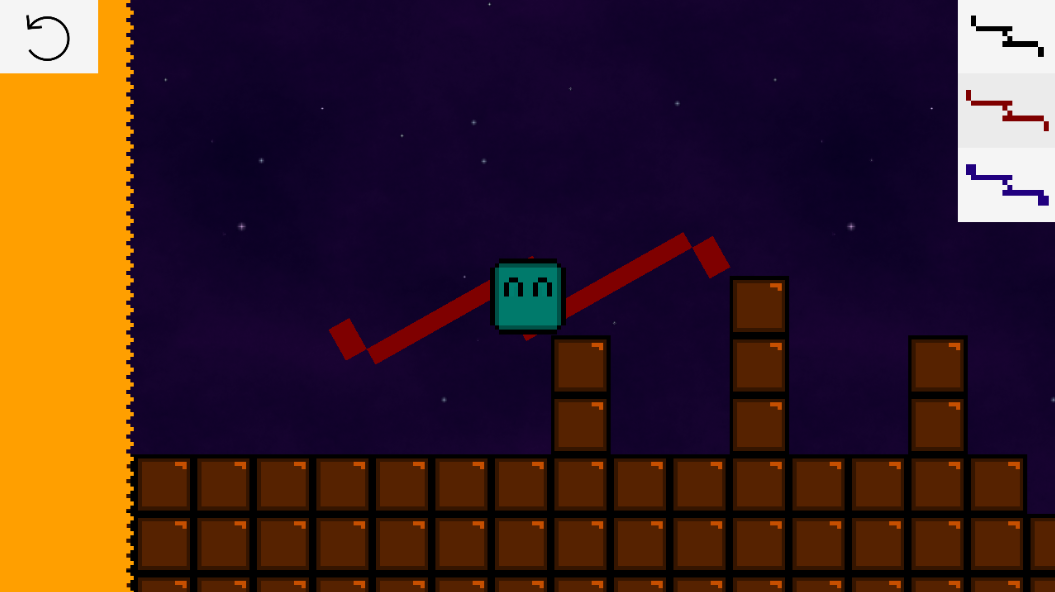
* Menu principal simples de exibição-exemplo;
* Fase inicial simples não desenvolvida de maneira detalhada para demonstração;
* Personagem-teste com 3 tipos de pernas giratórias diferentes;

**Tópico 1 – Jogabilidade e *Features***

BlockClimber 2D é um jogo simples desenvolvido para o teste para a vaga de Desenvolvedor C# Unity Júnior da VirtualHub e conta com apenas um menu/fase inicial simples para demonstração, antes de iniciar existem alguns fatos importantes a serem declarados:

* O requisito de desenho para conversão em *mesh* com colisor não pôde ser atendido devido à minha **junioridade** e por isso utilizei uma alternativa onde criei 3 opções de pernas giratórias dinâmicas para serem alteradas pelo jogador durante a *gameplay*.
* O jogo foi feito em 2D pois estou mais familiarizado em desenvolver mecânicas bidimensionais, isso não significa que não hajam conhecimentos com Unity3D (até porque o modo 2D do Unity é apenas uma planificação de seu modo tridimensional e para operar com o mesmo basta obter modelos 3D e utilizar Vetores 3 (XYZ) ao invés de Vetores 2 com um tipo de projeto 3D.
* Existem pequenas falhas na colisão das pernas com o cenário que podem ser corrigidas colocando um ***Rigidbody 2D*** nas pernas, porém ao perceber a interação que essas falhas promoviam eu particularmente gostei do efeito e resolvi manter – além disso inserir um corpo rígido nas pernas iria fazer com que elas se soltassem do corpo ocasionando em um bug.

**Como jogar:**

****

A gameplay é bem simples e intuitiva, ao iniciar o jogo e clicar/apertar no botão de iniciar você irá se deparar com 4 botões sendo os da direita opções de pernas (selecionar uma perna irá desativar as outras e as pernas são desativáveis) e o botão da direita um inversor de direção que poderá ser usado caso você fique preso(a) em algum lugar. O objetivo é simplesmente chegar ao final da fase sem tocar na parede de fogo que irá te seguir lentamente.

**Tópico 2 – Funções internas**

O jogo conta com funções internas bem básicas além da implementação de um asset gratuito chamado ***Simple Fade Scene Transition System*** que foi implementado duas vezes sendo a primeira para realizar a transição do menu principal para a primeira fase e a segunda para realizar a transição de morte para o início da mesma fase ao perder.

**Objeto da parede de fogo:**

O objeto da parede de fogo é relativamente simples, se trata basicamente de um script que altera a posição do elemento constantemente para frente unido a um script que faz o jogador perder (reinicia a cena) ao toque.

**Objeto do jogador/pernas:**

O Jogador não tem script para ele mesmo, ele apenas interage com a parede de fogo ao ser pego e contém as pernas em seu interior, as pernas contam com um script que as ativa/desativa conforme o input do usuário nos botões ou após ganhar a fase (onde todos os objetos são desativados, o som de ***level completed*** do Super Mario Bros é tocado e o jogo acaba).

As pernas contêm um script giratório que faz com que as mesmas girem para enganchar nas plataformas o que promove o movimento básico (**existem falhas de contato como citado anteriormente que poderiam ser resolvidas aplicando corpo rígido, porém devido às interações foi resolvido que fossem mantidas**).

**Botão de inverter:**

O botão de inverter basicamente altera a força aplicada no script que faz as pernas girarem multiplicando os valores por -1 (fazendo com que realizem sentido contrário).

**Objeto da câmera:**

O script que segue a câmera é simples, foi implementado um Vector3.MoveTowards para mover a câmera de maneira suave em direção do alvo (bloco azul que é o jogador).

**FAQ**

1. **Por que eu não utilizei prefabs?**

**Devido à simplicidade do projeto e aos poucos recursos que foram necessários para desenvolvimento da demonstração não achei necessário criar prefabs.**

1. **Por que criei o botão de inverter direção?**

**Apesar da utilidade baixa/ocasional o botão de inverter direção pode servir para casos onde o jogador fique preso em algum lugar devido a algum bug não planejado ou falha no level design (que não foi altamente desenvolvido por sem uma demonstração).**

1. **Por que não criei uma mecânica de desenho?**

**Costumo ser extremamente sincero e a resposta para isso é simples e rápida: eu não sei/não consegui implementar isso por enquanto. Até tentei realizar ações com o line renderer para mesh renderer porém não estou tão familiarizado com ambientes tridimensionais e vi que isso iria demandar muito tempo, então pensei em uma alternativa interessante que tivesse um peso semelhante.**

1. **Qual foi a arquitetura usada por mim?**

**Diante da simplicidade do projeto a arquitetura utilizada foi a de camadas onde a View (parte gráfica do jogo) se conectou apenas aos scripts simples sem sistemas prontos escalonáveis.**