

Um clone do famoso jogo de arcade *Space Invaders (1978)*

O Space Meteors é uma implementação em Unity3D do histórico jogo de arcade Space Invaders.

O Space Invaders é um jogo de 1978 inteiramente desenvolvido por uma pessoa, Tomohiro Nishikado, no qual foi produzido e vendido pela Taito no Japão.

Foi o primeiro *fixed shooter* e definiu o *template* para o género de *shoot ’em up*. O objetivo é derrotar a horda de inimigos que se movem horizontalmente e para baixo. O jogador é um canhão laser apenas pode se mover na horizontal no fundo do ecrã.

Nishikado afirmou que o jogo *Breakout* foi a sua inspiração para realizar o Space Invaders.

A versão original é single player mas com o passar dos anos foram criadas versões de 2 jogadores.

O jogadores está parcialmente protegido por *bunkers* estacionários que protegem o canhão de lasers inimigos.

Em relação aos inimigos, originalmente eram humanos a pilotar aviões mas estes foram alterados para naves aliens de forma a capitalizar no interesse da época sobre ficção científica relacionada com exploração espacial. Este género estava particularmente popular devido ao grande sucesso do 1º filme da saga Star Wars que estreou em 1977.

As naves inimigas têm o design de uma “lula” um “caranguejo” e um “polvo” inspirados no romance de ficção *The War of the Worlds* de H. G. Wells. Também existe uma nave que aparece aleatóriamente no topo do ecrã que ao acertar nela o jogador ganha uma quantidade de pontos aleatória.

Um dos atributos que ficou associado ao Space Invaders foi a curva dificuldade. Com o progredir do jogo e ao matar cada vez mais naves inimigas, as mesmas começam a mover-se mais depressa. Esta “feature” é na verdade um glitch não intencional causado pela dificuldade de processamento da máquina em processar todos os inimigos no início do jogo mas conforme existem menos inimigos, torna-se mais rápido o processamento e por consequência a frame rate aumenta consideravelmente.

O Space Invaders tinha um highscore pois na sua essência, é um jogo infinito e como tal um sistema de pontuações permitia a cada jogador deixar a sua marca e competir com outros jogadores pelo topo.

Em termos de tecnologia, os microcomputadores no Japão não eram poderosos o suficiente para realizar as tarefas involvidas no design e programação do Space Invaders e como tal, Nishikado teve que criar o seu próprio hardware e as ferramentas de programação para seu uso. Com isto, desenvolveu a sua própria arcade board utilizando compononentes recentes dos Estados Unidos. Os componentes integrados na sua criação foram um Intel 8080 (8-bit CPU), um monitor CRT com *raster graphics* e um *framebuffer* para os bitmaps bem como um sound chip da Texas Instruments e circuitos analógicos.

Sem qualquer sombra de dúvida, componentes bastante limitados e exponencialmente inferiores com os atuais. Estas mesmas limitações tecnológicas foram uma dificuldade que tornou o jogo ainda mais popular. Devido a estas limitações, o design foi aperfeiçoado e ainda hoje, mesmo com jogos fotorrealisticos a sairem todos os anos, o Space Invaders permanece no trono dos jogos históricos que moldaram a indústria. A própria soundtrack do jogo é composta por apenas 4 notas de baixo sintetizadas mas devido ao design inteligente de Nishikado, com o progredir do jogo a intensidade e velocidade destas simples notas recurrentes é aumentada de forma a jogar com as emoções do jogador.

3 no desenvolvimento, deve fazer uma reflexão técnica sobre a sua abordagem à implementação,

Relativamente a esta implementação, foram criadas as classes:

* Constants
* Enemy
* ExplosionDestruction
* GameController
* HomeManager
* MissileDestruction
* Player

Bem como os interfaces:

* IActions
* IControllable

De uma forma breve:

* A classe Constants serve para declarar todos os valores “tabulados” que dão jeito ter à mão para passar em métodos do Unity que requerem o nome do objeto/scene
* A classe Enemy tem a funcionalidade de disparo e colisão com os mísseis do jogador. Originalmente pensei em colocar a lógica de movimento nesta classe mas mais à frente foi pedido para essa lógica ficar dentro do GameController
* ExplosionDestructions serve para fazer destruir (Garbage Collecting) as animações de explosão
* GameController gere o movimento dos inimigos bem como Input do jogador para sair e Scene Loading entre Game e Home menu. Existem uma melhoria que faria a esta classe. Criava uma versão Singleton, pois foi-me bastante útil ter informação se a nave está morta ou se existem inimigos de sobra.
* O HomeManager é apenas uma classe para detectar Player Input e navegar para a Game scene.
* O interface IActions serve de contrato para os inimigos e o player. Como inicialmente desenhei a solução para o movimento dos inimigos ser autónomo à unidade mais granular, pensei em usar um interface para garantir que as mesmas eram aplicadas conforme as funções esperadas desses objetos. Foi também uma forma de praticar um bocado este modelo de programação.
* IControllable foi criado no pensamento de implementar um 2 player game. A sua função seria estabelecer o contrato para uma classe implementar métodos de deteção de player input
* MissileDestruction serve para detetar colisões entre o míssil do player e outro objeto. Sempre que este entra em colisão o mesmo deverá ser destruído, juntamente do objeto inimigo e instanciar uma explosão. Isto faz com que a gestão de explosão do meteoro e meteorito sejam tratadas no mesmo método
* Por fim, a classe Player gere o movimento, disparo, player Input, colisão com objetos e estados do mesmo. Inicialmente pensei em implementar um mecanismo de Spawn da nave mas acabei por não concluir visto não possuir muito tempo e não ser critério base.

Relativamente a dificuldades sentidas, diria que onde perdi mais tempo foi a tentar perceber porque é que com o Canvas/Background activos não era possível ver a minha nave. Após trinta minutos frustrados de incógnita e uma breve pausa pelo jardim percebi que tinha o Canvas em Render Mode “Overlay”. No fim de atravessar esse roadblock o projeto correu bem.

Ao longo do projeto criei um trello de forma a registar as melhorias que fiz e que acho interessantes para fazer.

Seguindo o enunciado do projeto, todos os Colliders2D seriam Box, mas visto que a nave tem uma forma poligonal não tradicional, os meteoros circulares e o míssil ser cilindrico, alterei para formas mais apropriadas sendo que para a nave, desenhei os pontos básicos do poligono com o PolygonCollider2D.

Também criei 2 prefabs para os inimigos visto que na spritesheet estavam 2 formas de rotação do objeto. Tal como no Space Invaders, existem diferentes tipos de inimigos. Tendo em conta que é um jogo sobre objetos espaciais naturais, seria interessante criar meteoritos de diferentes cores que ao rebentar, diferentes minérios seriam projetados na explosão. Nem todos os meteoros têm a mesma composição.

Vi que no enunciado o “disparo” dos inimigos seria para reutilizar o míssil do jogador. Achei essencial alterar este requisito visto que os meteoros naturais não têm canhões de laser.

No meu play-testing e debugging encontrei uma situação que fazia os inimigos lançarem uma chuva de meteoros sobre o jogador devido à lógica pseudoaleatória de disparo dos mesmos. Decidi contornar este comportamento indesejado ao criar uma “shooting zone” e “reload zone”. Em suma, os inimigos ao tentar realizar o disparo ou até mesmo fazer cooldown de disparos anteriores, têm que estar X coordenadas próximos da coordenada X da nave. Isto dá um aspeto de inteligência ao inimigo, pois parece que os eles tentam disparar quando a nave interseta o espaço de disparo.

Algumas features que fariam o jogo mais polido e interessante que eu listei seriam partículas de fumo e incandescência nos meteoros pois eles estão a atravessar a atmosfera e como tal aquecem por causa do atrito, partículas no burner da nave de forma a parecer que a nave está efetivamente ligada, um cometa a atravessar o topo do ecrã como se fosse a nave bonus do Space Invaders. Sendo um cometa, poderia deixar um trail de gelo subtil.

Uma implementação essencial seria um sistema de vidas e tornar o jogo com níveis infinitos em que o jogador navega fluidamente entre os mesmo. Ao atravessar cada nível o background “avança” visto que a nave está a “viajar para o espaço e a destruir cada vez mais meteoros”.

Com o jogo em modo infinito é necessário um Highscore System, pelas mesmas razões que o Space Invaders tem.

Algumas mecânicas que acharia interessantes de implementar seriam collectables dariam vidas extra ou aumentar temporariamente a fire rate ou velocidade da nave ou boosts que dariam habilidades aos disparos como taxa de penetração (1 míssil, 2 inimigos). Um conceito interessante seria também perks que ficam com o player até ao fim do jogo. Perks essas poderiam ser ativáveis como por exemplo, um “time warp” que faria camera lenta para tentar evitar colisões ou até mesmo mais fire power (2, 3 canhões laser).

Achei um projeto bastante educativo. Aproveitei para praticar os interfaces bem como, a nível de gestão pessoal, contei o tempo que demorei a fazer o projeto. Uma boa noção das nossas capacidades passa por entender quanto tempo demorariamos a realizar uma tarefa. À medida que ia fazendo o jogo comecei a ter ideias em basta e como tal decidi criar o Trello para tentar guardá-las todas para o futuro. Gostaria de concluír esta implementação em plenitudo com todas as features que enunciei bem como todas as demais que surgiriam organicamente ao trabalhar neste projeto.

Diogo Carvalho 11/08/2020