PROJETO

EPTA Entertainment

Neste documento encontra-se o plano de desenvolvimento que se refere à criação da versão 1.0 do aplicativo EPTA Space Program, desenvolvido pela EPTA Entertainment.

Desenvolvido por

Felipe J. O. Ribeiro
Mateus da Silva Fernandes
Olavo Caetano Inácio
Pedro Guilherme R. V. de Melo

Universidade Federal de Uberlândia

Equipe de Propulsão e Tecnologia Aeroespacial

EPTA Space Program - Versão 1.0

16 de janeiro de 2021

Resumo

Neste documento encontram-se as práticas adotadas no desenvolvimento da versão 1.0 do aplicativo Epta Space Program. O programa já havia sido lançado na google play em meados de 2017, mas ele foi retirado do ar devido a questões burocráticas (ausência de uma política de privacidade).

Nesta etapa de desenvolvimento a equipe atualizará o jogo para a versão mais recente do unity. Também revisará o work flow uma vez que na versão de 2017 estava impossível de escalar e inviável de manter devido a más práticas de programação. Como resposta a isso, a sub área se propôs a revisar todos os padrões de desenvolvimento de forma a estar conforme com as mais altas recomendações desta indústria.

As funções base do aplicativo são:

- TELA INICIAL Desenvolvimento da tela inicial com animações. Que não precise iniciar o jogo pausado.
- 3 FASES Três etapas distintas de jogo. (céu azul, extratosfera, espaço)
- FASE FINAL A certa altura (a determinar), o player tem a opção de de ir para a lua. (o conteúdo será adicionado posteriormente. Mas por hora o player recebe uma mensagem de congratulações)
- MORTE Som e efeito visual de explosão no momento de morte.
- ARCADE Mecânica base do jogo mantém-se arcade, com rolagem para cima com obstáculos e replay ágil.

As features que espera-se que estejam presentes no jogo na data de estreia são:

- PROPAGANDA As propagandas tipo banner e tipo Intersticial devem ser implementadas novamente.
- VERSÃO PAGA Deve haver uma versão sem propagandas e que é paga. (preço a determinar)
- TABELA INFO Tabela de informações sobre a equipe no menu opções.
- VOLUME Deve ser possível ajustar o volume.
- SALVAR A melhor pontuação do jogador deve ser salva localmente, assim como o volume.
- ASSESTS REUTILIZADOS Serão utilizados os assets do antigo lançamento. Novos devem ser feitos conforme necessidade.
- ALTURAS REAIS As alturas de mudança de fase devem ser consistentes com o mundo real.
- DIFICULDADE CRESCENTE O sistema de dificuldade deve ser implementado de forma a permitir razoável controle em função de fase e tempo.

A metodologia adotada para a criação do APP tem duas etapas importantes: Até o término da versão 1.0 será adotada uma metodologia em cascata, nesta que é a etapa de pré lançamento, onde se dividirão as features a implementar em uma ordem lógica de execução. Datas para pontos chave no desenvolvimento serão determinadas seguindo o padrão industrial (pre-alpha : 0.5.0, alpha : 0.6.0, beta-1 : 0.7.0, beta-2 : 0.8.0, ..., release : 1.0.0), onde cada uma terá um objetivo e uma lista de requerimentos a serem atendidos. Em seguida, após lançado, será adotada a metodologia ágil para desenvolvimento contínuo de atualizações e melhoramentos conforme necessário. Esse será chamada de etapa contínua de pós lançamento.

Será implementado um ciclo de desenvolvimento completamente integrado à ferramente Git de controle de versão e à página GitHub (https://github.com/Epta-space/EPTA_SPACE_PROGRAM), onde o jogo será salvo em cloud em um repositório privado. Assim, essas ferramentas servirão para possibilitar o desenvolvimento em grupo do código fonte, além de também ser uma forma de quantificação da produtividade, uma vez que o controle do número de commits é uma métrica importante no acompanhamento das atividades. Mais detalhes podem ser encontrados na descrição das etapas de desenvolvimento do pré lançamento.

A criação do produto mínimo viável já foi feita na primeira versão do jogo, motivo pelo qual não temos a etapa MVP neste texto. Apesar disso será inserida aqui alguma documentação sobre esse processo nos capítulos iniciais. O desenvolvimentos que ocorreram anteriores a esse relatório são o motivo, também, da enumeração das etapas começarem em 0.5. Nos capítulos iniciais também será dada uma detalhada noção do estado atual do aplicativo assim como algumas informações sobre como foi o desenvolvimento até então. Tudo que foi feito até o momento foi desenvolvido durante o ano corrido de 2020, com exceção dos assets que vem da versão anterior.

Dentre as informações do estado atual do aplicativo serão listados os scripts que existem atualmente em formato UML e por escrito, a estrutura de arquivos no projeto unity e a estrutura de Prefabs. Estas estruturas também já atendem a padrões de projeto sugeridos online e facilitam o desenvolvimento do controle de versão via Git, além de organizar as informações para facilitar o acesso e aumentar a escalabilidade.

Próximo ao fim do documento também serão apresentadas algumas propostas de melhoramento na etapa ágil de lançamento, isso é, após a primeira versão do programa ser lançada na Google Play. Isso mostra as possíveis rotas que a subárea tomará no futuro, após o começo da etapa de desenvolvimento contínuo.

Palavras chave: Desenvolvimento de Software, aplicativo, android, jogo, google play.

Sumário

Lista de Figuras

Capítulo 1

O desenvolvimento até aqui

O início deste projeto ocorreu no começo do primeiro semestre de 2018. Um grupo de membros se juntou no desenvolvimento do aplicativo como uma nova proposta de angariação de fundos para a equipe. Observou-se o potencial do mercado de aplicativos global, que cresce consistentemente ao longo dos anos. Além disso pensou-se na oportunidade de se aprofundar no desenvolvimento de software e o aprendizado que isso traria.

Ao longo do ano o aplicativo foi desenvolvido e finalmente lançado em dezembro de 2018. Ele foi baixado em mais de 40 países e mais de 800 vezes, com uma avaliação média de 4.88 estrelas de 5.

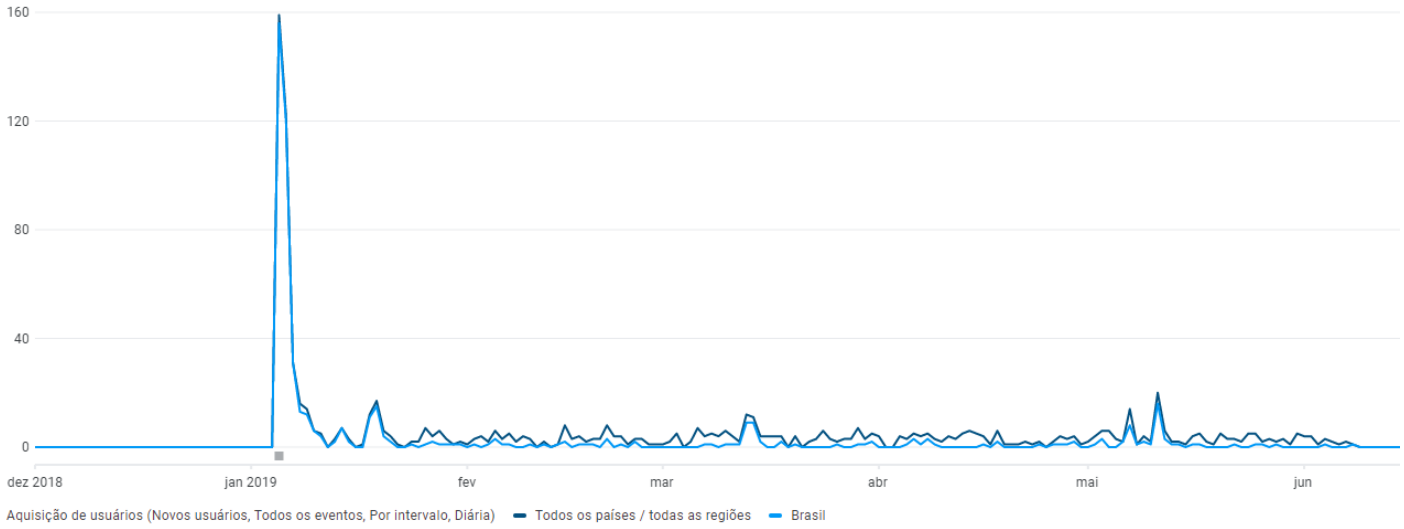

Fig. 1.1: Downloads do aplicativo distribuídos no tempo.

Em julho de 2019 o aplicativo foi retirado da Play Store devido à ausência de uma politica de privacidade. Isso deu início a um longo hiato no desenvolvimento que só foi retomado no início de 2020.

A estratégia de monetização usada foi a de mostrar propaganda a partir dos serviço AdMob da Goggle. No tempo que o aplicativo ficou disponível para download (5 meses), somente com publicidade exibida durante as partidas, foi possível angariar 14 dólares. Não foi possível resgatar o valor, uma vez que só se pode fazer isso em pacotes de 100 dólares.

Considerando que o aplicativo só ficou disponível por 5 meses e que estava em seu estado base, sem qualquer atualização ou melhoramento, considera-se este retorno um ótimo resultado. Espera-se ultrapassar essa marca com o novo lançamento em 2021.

1.1 Sobre o papel do desenvolvimento de jogos na equipe

Este é um assunto de grande complexidade.

Na ciência da computação, o assunto é tratado como um dos mais difíceis, uma vez que, assim como na criação de foguetes, é uma prática extremamente multidisciplinar. Dentre as competências que se espera de uma equipe que se propõe a tal papel temos todo tipo de habilidade que oscila entre técnico e artístico de forma dramática.

Tendo a diversidade do assunto em vista, observa-se uma oportunidade de uma rica interlocução de agentes das mais diversas áreas, assim como vemos de um ponto de vista geral, dentro da equipe de propulsão e tecnologia aeroespacial. Desse forma espera-se que a subárea de desenvolvimento de jogos usufrua dessa diversidade que existe dentro da equipe, e até a expanda ainda mais.

Outro aspecto importante desta prática é a forma como se enxerga a programação. Na pesquisa e na engenharia também é constante a prática de programar, mas isso é feito de forma pragmática e apressada. Esse é o natural, uma vez que o código não é o produto final, mas um meio que se precisa percorrer, seja para uma pesquisa ou para desenvolver uma estrutura ou peça para o foguete. Tudo isso é familiar aos outros integrantes da equipe e para qualquer um da área de engenharia. Mas, dentro da EPTA Entertainment a perspectiva é outra. O código que está sendo desenvolvido é o produto final, e como tal, deve ser feito de forma bem diferente do código acadêmico. Além de funcionar ele deve ser legível e obedecer a padrões de projeto. Além disso, no lugar de exercer uma funcionalidade única e específica, códigos como os desenvolvidos na subárea muitas vezes precisam ser flexíveis e robustos de forma a trabalharem harmonicamente no ecossistema lógico do aplicativo.

Desenvolver essa outra perspectiva de desenvolvimento de software é extremamente positivo a qualquer engenheiro, uma vez que mais e mais o engenheiro moderno se vê cercado por programação, em qualquer especialidade.

1.2 O Mínimo produto viável(MVP)

O jogo para celular "Epta space program"é um programa desenvolvido para Android cujo público alvo são os entusiastas da indústria aeroespacial. Desse forma, o objetivo da equipe é criar algo acessível e informativo.

Capítulo 2

Estado do aplicativo

O aplicativo já se encontra em um estado avançado de desenvolvimento, consequência dos anos de trabalho que se foram. Nesse capítulo será descrito com ele se encontra no início de 2021.

Dentre as coisas descritas temos os seguintes tópicos:

- Disposição dos arquivos.
- Disposição de Game Objects.
- Ecossistema lógico.

2.1 Disposição de arquivos

O gerenciamento de arquivos é muito importante no desenvolvimento de jogos.

O objetivo é organizar a informação, uma vez que existem muitos arquivos (centenas). E podem ser figuras, música, scripts, meta dados, e objetos lógicos de jogo, como prefabs, animações, cenas e muitas outras coisas. Dessa forma é muito importante que tais arquivos sejam dispostos de uma forma inteligível, uma vez que mexer neles faz parte do cotidiano dos desenvolvedores.

Existem dois diretórios importantes que devemos saber a composição. O diretório Assets e o diretório Adm. Ambos estão localizadas no diretório principal do projeto Unity do aplicativo.

No primeiro se coloca tudo que entra no desenvolvimento do jogo. Entre as coisas que estão lá dentro temos os scripts, músicas, prefabs, etc... Toda vez que se quer modificar algo no jogo, um dos arquivos dentro desta pasta será modificado.

No diretório adm são colocados arquivos que não são usados no desenvolvimento do jogo, mas ainda sim são importantes. Entre as coisas presentes lá temos o relatório que documenta a criação do aplicativo (como este que estão lendo), a política de privacidade que usamos no lançamento do aplicativo, a chave de publicação, e outros detalhes.

Dentro do diretório Assets existe uma estrutura interna de pastas que dispõe recursos utilizados no desenvolvimento de forma organizada e escalável. Existes recursos gerais e recursos específicos de fases, eles são dispostos da seguinte forma:

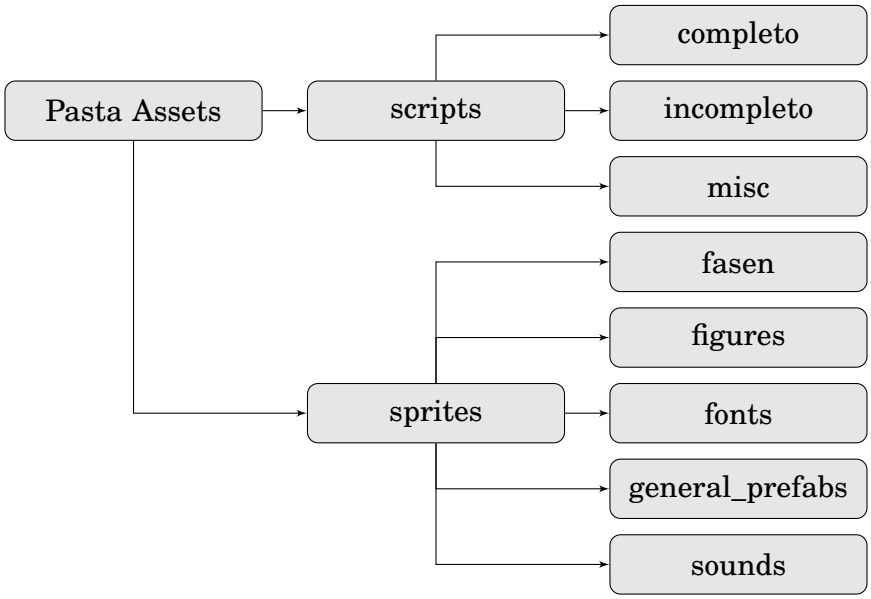

Fig. 2.1: Sistema de arquivos da pasta Assets. O que fora ignorado não tem importância para o desenvolvimento atual.

Dentro da pasta scripts temos três diretórios. Um para programas em c# completos, ou seja, já funcionais para o que se propõem. Outro para programas incompletos (coisas que estão em desenvolvimento devem ser postas aqui, até segunda ordem). E uma terceira pasta para scripts em geral, ou seja, para testes de alguma funcionalidade.

Dentro do diretório sprites encontram-se todo tipo de asset utilizado no jogo. Nas pastas fonts, general_prefabs, sounds e figures estão assets de uso geral no jogo, enquanto nas pastas fasen estão assets específicos de alguma fase.

- fasen: Assets específicos de uso na fase de número n. Aqui encontra-se os prefabs dos obstáculos e os componentes do background dinâmico.
- figures: estão assets visuais de uso geral, como as logos das universidades, os assets do background dinâmico inicial, a skin do player e elementos visuais da UI.
- fonts: encontram-se todas as fonts utilizadas no aplicativo.
- general_prefabs: encontram-se prefabs relacionados a UI e aos componentes puramente lógicos do jogo, cada qual em sua pasta dedicada (UI_prefabs e Game_rules).
- sounds: Nesta pasta encontram-se as músicas e sons do jogo.

Diretamente dentro da pasta Assets também encontra-se o arquivo "Player.Scene"que é a única cena do jogo. Nela encontram-se todas as referências a todos os elementos exibidos dentro do jogo.

2.2 Disposição de Game Objects

A plataforma Unity permite a divisão e objetos lógicos do jogo em "Game Objects", estas entidades são unidades de informação. Nelas é possível se armazenar qualquer tipo de elemento que possa aparecer

dentro do jogo, inclusive outros "Game Objects". Tal procedimento permite dividir os elementos lógicos da cena em unidades funcionais. Essas unidades podem então ser salvas de forma não volátil nos chamados "prefabs".

Dessa forma, dividir em unidades funcionais os elementos lógicos do jogo e separá-los fisicamente na memória do projeto resulta em muitos benefícios. Um deles é a organização e a escalabilidade. Torna-se muito fácil referir-se e um elemento do jogo quando a estrutura lógica é bem feita. Basta dar as coordenadas do elemento a partir dos Game Objects que a englobam. A estrutura ramificada garante um ambiente organizado mesmo com centenas de elementos na tela.

Outra vantagem está no controle de versão. Quando esses elementos lógicos são divididos por funcionalidade de forma física na memória, isso significa que ao se alterar elementos referentes a uma unidade, essas alterações estarão confinadas aquela unidade, não interferindo em outras. Tal propriedade permite que muitos integrantes da equipe de desenvolvimento trabalhem simultaneamente sem que editem os mesmos arquivos ao trabalharem em áreas diferentes do aplicativo. Isso facilita o processo de unificar o trabalho dos integrantes da equipe e garante menos problemas na utilização de sistemas de controle de versão como o "Git", ferramenta utilizada pelo grupo.

A seguir encontra-se a estrutura destes Game Objects na cena principal de jogo:

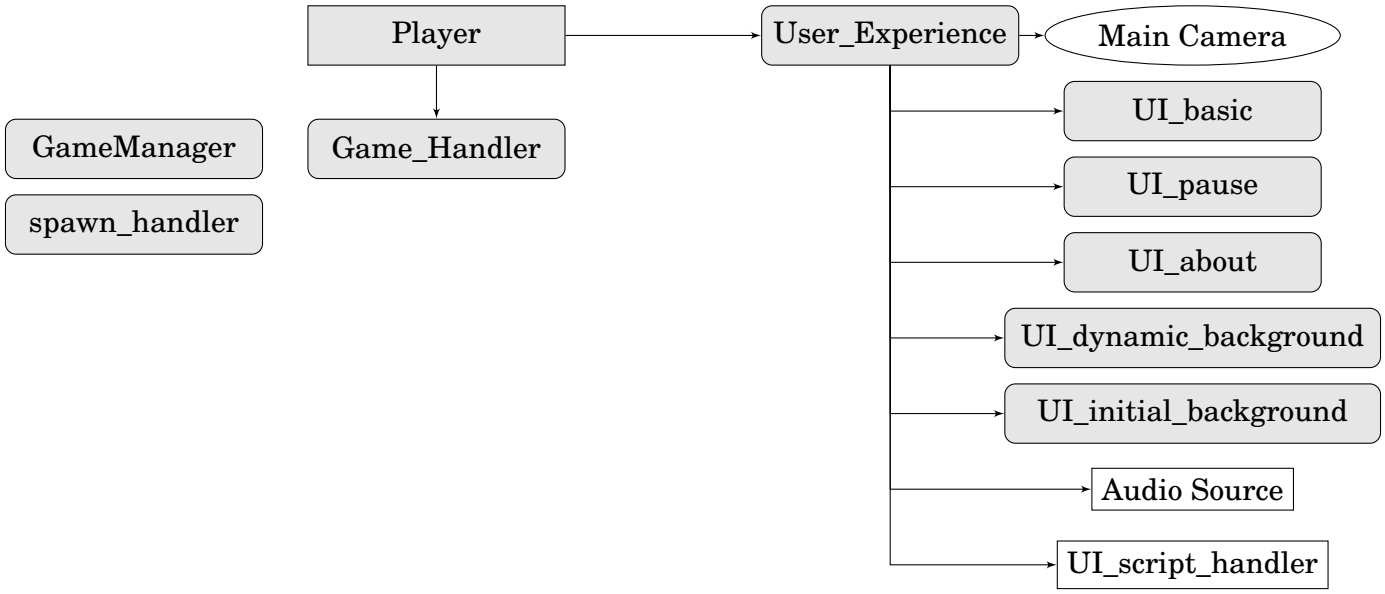

Fig. 2.2: Árvore de objetos de jogo, prefabs em azul.

Além dos elementos mostrados acima, temos também o player e o objeto "EventSystem". O desenvolvedor que precisar alterar estes elementos é o única que terá a liberdade de fazer alterações no arquivo "Player.scene". Qualquer outra alteração será feita nos dois prefabs principais ou nos outros que estejam dentro destes.

2.3 Ecossistema lógico

```
using UnityEngine;

public class Destroy_obstacles : MonoBehaviour
{

     // Destroi tudo que encostar neste objeto
     void OnCollisionEnter2D(Collision2D coll)
     {
         Destroy(coll.gameObject);
     }
}
```

Fig. 2.3: Código do script "Destroy_obstacles.cs".

```
using UnityEngine;

public class audio_controls: MonoBehaviour
{
    // Reference to the scene audio controll
    private GameObject music_volume_control;

    // makes the starting volume change in UI
    [SerializeField]
    private float volume;

    // Start is called before the first frame update
    void Start()
    {
        // Acha o game audio principal, com a música do jogo via tag (
            GameObject Audio Source)
        music_volume_control = GameObject.FindWithTag("audio_control");

        // Declara o volume
        music_volume_control.GetComponent<AudioSource>().volume =
            volume;

    }

    // Quando o valor do volume muda, atualiza o valor no Audio source
        da música
    public void volume_controll(float volume){
        music_volume_control.GetComponent<AudioSource>().volume =
            volume;
    }
}
```

Fig. 2.4: Código do script "audio_controls.cs".

Capítulo 3

Plano de gestão para 2021

Tabela 3.1: Planejamento escalonado

Recurso	Semana 1 (01/16)	Semana 2 (01/23)	Semana 3 (01/30)	Semana 4 (02/06)	Semana 5 (02/13)	Semana 6 (02/20)	Semana 7 (02/27)
Background dinâmico							
Memória não volátil							
Tela inicial dinâmica							
Recursos de fim de jogo							
Mecânica de Spawn							
Propaganda funcional							

Capítulo 4

Planos futuros