**Technical Design Document**

**Team4**

**Pango Blocks! - Remake**

# **Introduction**

Creation of the remake Pango Blocks! By Studio Pango.

# **2 Purpose of this document**

This is a Technical Design Document that explains the technical aspects of this remake designed and developed using the Unity engine in detail. This will give an overview of the technical design of the project and can be used by developers to understand the code workflow and logic.

# **3 Scope**

**The scope of this document includes:**

• Environment Specification  
• System Requirements  
• Prerequisites  
• File Folder Structure

• Code Structure Design

*Note – this is a Technical Design Document which only covers the technical aspects. Please refer to the Game Design Document for any other information about the design processes*

# **Environment Specification**

The project was developed in the **Unity Engine; Version 2022.3.10f1 LTS**.

The project include the **TMP** framework.

The project also use an external code base framework “**VUDK**”.

The project use multiple **notable** unity components.

Used **UI** Unity Components:

• Canvas

• Canvas Scaler

• Graphics Raycaster

• Rect Transform

• TMP-InputField

• TMP-Text

• TMP-Button

Used **Physics** Unity Components:

• Rigidbody 2D

• Polygon Collider 2D

• Box Collider 2D

Used **Sequencing** Unity Components:

• Playable Director

• Signal Receiver

In the scenes *“scn\_DebugGameScene”* and *“scn\_DebugCutsceneGame”* was developed and debugged the code for the game.

# **5 System Requirements**

The Developer System (specifications below) was used to develop this Technical Design Document. It is important to note that the system the game is running on should also have similar specifications to ensure a proper functioning of the game.

The system specifications for Pango Blocks! - Remake process:

|  |  |
| --- | --- |
| **Operating System** | Android 11.0.0 |
| **Processor** | Snapdragon™ 460 |
| **RAM** | 4 GB |
| **Hard Disk** | 64 GB |
| **Display** | 90Hz 6.52” 1600 x 720 (HD+) |
| **Phone** | Oneplus Nord N100 - BE2013 |

# **6 Prerequisites**

The prerequisites for the robot to successfully run are as follows:

1. The game APK application is installed in an Android operating system of version 5.1 or higher.  
   b. The system has sufficient disk space of 100 MB to install the APK and manage its save data.

c. The system should have similar specifications.

# **7 File/Folder Structure**

**• Assets Folder:**

Path at which all the assets developed are stored.

**• Art Folder:**

Path at which all the art assets are stored. Contains other folders.

1. Animation Folder:Contains all the animations assets.
2. Fonts:Contains all the fonts.
3. Materials: Contains all the materials.
4. Models: Contains all the 3d models.
5. Sprites: Contains all the sprites.
6. Textures: Contains all the Textures.
7. VFX: Contains all the VFXs.

**• Audio Folder**

Path at which all the audio assets are stored. Contains other folders.

1. Mixers: Contains all the Mixers used in the project.
2. Music: Contains all the audio clips relative to the music.
3. SFX: Contains all the audio clips relative to the effects of the game.

**• Code Folder**

Path at which all the code of the project is stored.

**• Docs**

Path at which all the documents are stored.

**• Level**

Path at which all the data relative to the game is stored. Contains other folders.

1. Data: Contains all the scriptable objects of the project.
2. Prefabs: Contains all the prefabs of the project.
3. Scenes: Contains all the scenes of the project.

|  |  |
| --- | --- |
| File/Folder | File/Folder Location |
| Assets Folder | Assets/ |
| Art Folder | Assets/Art/ |
| Audio Folder | Assets/Audio |
| Code Repository | Assets/Code |
| Docs Folder | Assets/Docs |
| Level Folder | Assets/Level |

# **8 Code Design**

8.1 Used Design Patterns

* ***State Machine***
* ***Observer***
* ***Singleton***
* ***Object Pool***
* ***Concrete Factory***
* ***Abstract Factory***
* ***Factory Method***

8.2 Code Structure

The code is structured using the singleton pattern for the MainManager class that manages all the fundamentals managers and specific managers for the game.

*8.3 Notable Classes*

Below are listed and described all the most important code for the *core mechanics* of the game. *Read the code summaries for more info about the code.*

***Managers***

* BlocksManager: This manager manages all the interactions of the blocks.
* BlocksDragger: This class manages to control the drag aspect of the blocks.
* PBRGameMachine: This is the state machine the controls the workflow of the level gameplay.
* GameStats: Manages all the important stats of the game.
* GameSceneManager: Manages the loading of the correct scenes.
* GameUIManager: Manages all the relative UI components.
* ProfileManager: Static class responsible of the creation, modification and deletion of profiles.

***Blocks System***

* BlockBase: Base abstract class for all the blocks.
* PlaceableBlockBase: Base abstract class for all the place-able blocks.
* SInglePlaceableBlock: Class for place-able blocks that are composed by one unit, such as the square or the triangle.
* ComplexPlaceableBlock: Class for place-able blocks that are composed of multiple SinglePlaceableBlock, such as the Rectangle or Trapezoid.

***Grid***

* LevelGrid: Manages the creation of the grid using its LevelTile.
* LevelTile: Class used for the LevelGrid for its tiles, that is responsible of the occupation of a tile and its properties.

***Path System***

* PathCalculator: Static class that manages the calculation of the correct path.
* Path: Class that store the data of a calculated path, such if is arrived at the objective or not, and an array of Nodes.
* Node: Class that store the information of each node of the path, such as the node position.

***Player***

* CharacterManager: Manages all the components of the character.
* CharacterPathRunner: Class responsible of the movement of the character through a Path.
* CharacterGraphicsController: Class responsible for all the graphic aspects of the character, such as its animations.

# **10 Dependencies**

• UnityEngine;

• UnityEngine.UI;

• UnityEngine.SceneManagement;

• UnityEngine.Events;

• TMPro;  
• System;  
• System.LINQ;  
• System.Collections.Generic;

• VUDK.\*; *\* - multiple dependencies from this framework, read the code for more info.*

• ProjectPBR.\*; \* - namespace used for the project, read the code for more info about the structure.