**Game Developer Task**

**Jelly Smash:**

**Match 3 game like Jelly Splash**

Contents

[Folder Structure 2](#_Toc10418105)

[Screens and Popups Used In Game: 2](#_Toc10418106)

[Approach Used: 2](#_Toc10418107)

[Files Used: 3](#_Toc10418108)

[How Screen Works? 3](#_Toc10418109)

[How Popup Works? 3](#_Toc10418110)

[How Map Screen is Created? 3](#_Toc10418111)

[How Map Level Works? 4](#_Toc10418112)

[How Objects are created? 4](#_Toc10418113)

[How Game Level Works? 4](#_Toc10418114)

[How connection of jellies work? 4](#_Toc10418115)

[How Score works? 4](#_Toc10418116)

[Use of DoTween: 5](#_Toc10418117)

[Use of Sound: 5](#_Toc10418118)

Jelly Smash

## Folder Structure

JellySmash

1. App Assets
   1. Fonts
   2. Library : Contains scriptable objects to store game data
   3. Prefab
   4. Sound
   5. Sprites
2. App Assets Code : Contains code for all App Assets
   1. Factory
   2. Library
   3. Prefab
3. Extensions : Scripts that are used for extended functionality
4. Model : Stores application data
5. Presenter : Stores all presenters used in application
6. View : Stores all views used in application

## Screens and Popups Used In Game:

1. Screens
   1. Home
   2. Map
   3. Play
2. Popups
   1. PlayLevelPopup : This popup appears when a map level is clicked
   2. GameOverPopup : This popup appears when game is over i.e. either player wins or loss.

## Approach Used:

Model , View , Controller (Presenter), Properties.

## Files Used:

1. Properties : This file is used to store all public variables. These variables are mostly set from inspector.
2. PrefabBehaviour : This is parent class for all Properties. An Object set with properties as child of Prefab Behaviour gets 2 files added to it at runtime. Files added at runtime are : View and Presenter of same element.
3. View : This is used to access Properties and Presenter data and show it on the UI. It takes data from Presenter and sets it in UI. All UI related code is done in view.
4. ViewBehaviour : This is parent class for all Views. ViewBehaviour takes 2 files Properties and Presenter and then the property and presenter are referred as Prefab and Presenter respectively in the view.
5. Presenter : This is used to access data from model, listen Actions triggered from other presenters, provide data to view.
6. PresenterBehaviour : This is parent class for all Presenters.
7. Model : This is used to store data.
8. Factory : This is used to create objects according to data provided to it.
9. Library : These are scriptable objects that contain details for screen, level, maps, etc.

## How Screen Works?

All screen are created under Screen object in canvas. Screen object has ScreenProperties. At run time ScreenView and ScreenPresenter are added to it. ScreenView and ScreenPresenter handles creation of all screens.

Initial Screen is created from ApplicationLoader at start of Application.

## How Popup Works?

All popups are created under Popup object in canvas. Popup object has PopupProperties. At run time PopupView and PopupPresenter are added to it. PopupPresenter handles creation of all popup.

## How Map Screen is Created?

There are 2 library for map screen:

1. MapLibrary : It contains prefab of all maps.
2. LevelLibrary : It contains details of all levels.

Each Map prefab has its levels position set in prefab. At run time level objects are created at the positions set in child of Map prefab.

Map are created from Map Factory and Levels are created from LevelFactory.

## How Map Level Works?

On click of Map Level data is saved into Game Model for current selected level. Then Play Level popup is shown with details of selected level. On Click of Play button, Game Play screen is loaded.

## How Objects are created?

On Start function of view we create objects of Prefabs that need to be visible in the view. Like we create tiles and jellies for grid on Start function of its View using data from presenter or properties. To create object from prefabs we have created different factory files.

## How Game Level Works?

On Map, when a level is clicked it is saved as current selected level in Game Model.

When play screen is created, on Start of PlayScreenView we create grid according to current level details taken from levels library. On creation of grid, we create tiles for the grid according to grid size i.e. rows and columns set in level library.

After grid is created, we create jellies to grid using UIGrid, and fill all tiles with the jellies. Jelly colors allowed per level is set in Level Library. According to information in level library for current level, jellies are created.

## How connection of jellies work?

JellyView check for OnMouseDown, OnMouseEnter and OnMouseUp. If OnMouseDown is there then selection is started, on mouse enter tiles are checked whether they are valid with respect to already added jellies or not. Valid jellies are added to selected jellies list. OnMouseUp, if there are 3 or more than 3 valid selected jellies then we remove those jellies and provide score for jellies to the player. Then, current jellies are moved down to fill grid downward and for empty tiles, new jellies are created.

## How Score works?

We have set score for first 3 tiles and all other tiles in Grid Model. Suppose if 5 tiles are removed, then for first 3 tiles each get 200 points and all other gets 250 each.

Total score for 5 tiles => 200 + 200 + 200 + 250 + 250.

50 is Additional Tiles Score and after each 3 tiles score gets on added by 50 for example for 9 tiles score is:

* (200+200+200) + (250+250+250) + (300+300+300)
* 600 + 750 + 900
* 2250 points

## Use of DoTween:

DoTween has been used to move jellies down, to create new jellies and move them to desired tile positions and for score text scrolling effect.

## Use of Sound:

Added sound for:

1. Background Music
2. Button Click
3. Win
4. Loss

For playing sound, AudioComponent Extension has been created to play sound.