Folder structure

- Art - source art and texture atlas made by hand in photoshop
- Bin - compiled binaries for Android, Win and OSX
- Docs - Instructions + this file
- Unity - Unity project

Unity project

There is only one scene file located in *Assets/Game/main.unity.*

The scene contains basic data structure:

- Base scene object is *Field*, which has attached GameManager and ArtData. GameManager is essential for starting up the game. ArtData script contains pointers to gameobjects and some other tweakable art-related values.
- *Center* game object is the root for the gem field.
- *Mask* is for obscuring falling gems from top.
- *Timer* is animated spark indicating the remaining time.
- *EndScreen* is shown when the time is up, with score and retry button.
- *Prefabs* contains gem gameobjects for instantiation.

Scripts

All Scripts are located in *Assets\Game\Scripts*.
It is somehow documented, although not too much. Code should be self-explanatory.

There are three main components: *Logic, Visual* and *Simulation*.

- *Logic* contains game state with the game field and performs basic game related operations like matching and collapsing of the gem field.
- *Visual* contains visual representation of gems and performs animations based on output from Logic.
- *Simulation* gets the inputs from the user and propagates them to Logic and Visual components. It also features the main game loop.

For the purpose of this simple test it could seem as a waste of time to create such a structure, but it will pay off once there is a need to expand the functionality.

The complexity of code is also increased by the ability of matching gems in parallel to collapsing the board, so the player doesn't have to wait for gems to animate. It helps the smoothness of the gameplay significantly.

All the art definitions are located in ArtData script. All the gameplay related definitions are in the LevelData script, both in the root of the project. It helps the extensibility of the game, once the new gem types will be introduced.

ArtData are exposed in the Unity editor, so artists can edit them easily (therefore the public members).

LevelData are a bit complicated in structure. It should be edited in some custom Unity editor extension, which I have omitted for now.

have used some of my “reusable” code - it is in the *Tools* folder and it was not written while creating this test.

Scripts handling end game screen and timer are located in the *Ui* folder.

Score is counted in very simple way - one matched gem increases score by 1.

There is a prefab system for gems in place, so they can be customized and animated easily: instead of just swapping sprites on the gem objects, different types of gems connects as a children to the gem object in the hierachy.

There is a basic optimization for reusing the gameObjects of various types of gems. Changing gem type means putting current visual representation hierarchy into the cache and asking the cache, whether there is available new gem type visual representation hierarchy. If it is not available, cache instantiates new visual representation. It is handled in the Cache script.

There can certainly be more optimizations in place (for example disappearing the gems will create new visual instance, some script arrays could be reusable etc.)

I haven’t implemented any type of dynamics for animation of falling gems for the sake of project simplicity.