Dream Games
Software Engineering Study

In this study, you are expected to create a simple, level-based mobile game. Implementation should be done in **Unity** and **C#.** The game must be playable on Unity Editor.

Flow

- When the user launches the game, s/he will see a **LevelButton** in the **MainScene.** LevelButton displays the current level of the user. When all levels are finished, the LevelButton should display finished text.
- The background of the MainScene should be an area image.
- After tapping the LevelButton**, LevelScene** should be loaded according to the current level. (You can find gameplay details in the Gameplay section)
- When a user wins the level
    - Celebration particles and animation should be shown to the user
    - MainScene should be loaded
- When a user loses the level
    - Fail popup should be shown to the user which has options to return to MainScene with a close button and replay the level with a try again button.
- The level progress of the user should be persisted locally. When Unity is restarted, the game must continue from the last saved state.

Level 1

All levels are finished

Gameplay

The goal is to clear all obstacles by blasting cubes near them or using special items.

Mechanics

- There is a rectangular grid where each cell can have one item. The grid can have a width and height of six to ten cells.
- There are four types (colors) of **Cubes**. At least 2 adjacent (non-diagonal) cubes with the same color can be blasted by tapping any cube in the group.
- All cubes fall down to empty cells only in the vertical direction. New random cubes should fall from atop the grid. They should come from the pool instead of new instantiation. Fall mechanics should be implemented by coding, not Physics or Unity animations.
- A move is spent after a valid tap operation.
- The level will be won if all obstacles are cleared within a given move count.

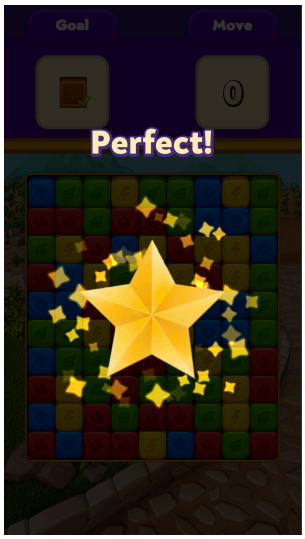
Special Item

- Special items will be created at the location of the tapped cube.
- Special items will explode when tapped or get damaged by another special item.
- Special items will fall down to empty cells only in the vertical direction.
- Cube groups should display a special item logo on them if they are eligible to create one.
- **TNT** will be created when the blasted cube group count is bigger than or equal to 5. It will explode in a 5x5 area.
- A TNT will be combined with other adjacent TNTs to create a combo when tapped. TNT-TNT combo will explode in a 7x7 area.
- **Rockets** will be created in horizontal or vertical directions randomly when the blasted cube group count is bigger than or equal to 3.
- A Rocket will be combined with other adjacent Rockets to create a combo when tapped. Rocket-Rocket combo will explode in both directions.
- A TNT will be combined with other adjacent Rockets to create a combo when tapped. TNT-Rocket combo will explode in both directions with a length and width of 3 cells.

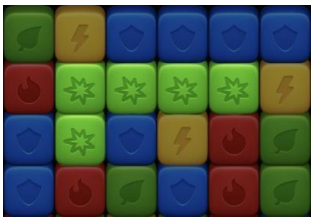
Obstacle

- Obstacles are goals of levels. Users should clear all obstacles to win the level within a given move count. There are three types of obstacles.
- **Box** takes damage when a blast occurs in adjacent cells or a special item explodes within range. Box can be cleared with one damage. Box does not fall down to empty cells.
- **Stone** takes damage only when a special item explodes within range. Stone can be cleared with one damage. Stone does not fall down to empty cells.
- **Vase** takes damage when a blast occurs in adjacent cells, or a special item explodes within range. The vase can be cleared with two damages. Vase takes no more than one damage from a single blast. Vase falls down to empty cells only in a vertical direction.

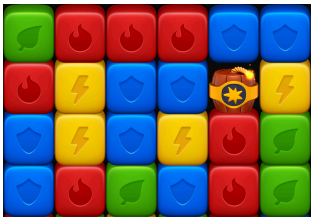
Board with all items



Celebration



TNT Hint



TNT Created on the clicked cell

Levels

All 10 levels should be included in the project to make them available to users for the initial launch. The level file structure is as follows;

- **level_number**: number of the level
- **grid_width**: width of the grid
- **grid_height**: height of the grid
- **move_count**: given move count for the level
- **grid**: a list of items (**r**: Red, **g**: Green, **b**: Blue, **y**: Yellow, **rand**: One of random colors(r, g, b, y), **t**: TNT, **rov**: Vertical Rocket, **roh**: Horizontal Rocket, **bo**: Box, **s**: Stone, **v**: Vase) starts from the **bottom left** of the grid and continues horizontally, ends at the **top right** of the grid.

Notes

- **Unity 2022.3.8** and **C#** should be used in the implementation with Unity's built-in renderer.
- Editor menu items should be implemented to adjust the current level.
- You can use third-party **tween libraries** for animations.
- You shouldn't use third-party dependency injection libraries like Zenject.
- The game should support portrait (9:16) orientation.
- All necessary assets are given in the case study folder.
- Try to make nice animations as best as you can. You can analyze Royal Match for animations.

Submission

You need to submit a Unity project which is playable from the editor. You need to share a private GitHub repository with software.engineering.study@dreamgames.com .