Computing Individual Project Report

Match 3 Game Framework in Unity*

Tomáš Kimer†

December 19, 2013

1 Introduction

As a one-term[1] Computer Games Development Erasmus student from Brno University of Technology, one of my study modules was the Computing Individual Project (SE3U608). The project was not intended to follow the traditional module scheme, because the original module is for the whole academic year. Instead of that, the project had a practical form of collaboration with CEMAS[2] on their game projects for mobile platforms. It was following the Prof Andrew Ware's[3] advice, because my originally requested Computer Games Development modules were not available for the one-term only study period.

I was in contact with the senior CEMAS programmer, Dann Rees[4], who tasked me with starting the development of the framework for a "Match 3" type game (similar to Bejeweled/Candy Crush) in Unity. But because there were some constant problems with getting the specific NDA forms for me to sign, I couldn't go through the design document, so the task was only a general framework with no specific/original characteristics.

This report briefly describes the design, implementation and results of the Match 3 type game framework in Unity.

2 Design

The project was designed in the two main iterations. The first one included a general gameplay framework with the following features:

- Basic Match 3 gameplay logic – valid game board generation, valid swap checks, detecting and destroying matches, score counting, performing drops and generation of new items, checking for next possible swaps (if zero, game is over or the board needs to be re-generated).
- Variable sized rectangle game board, with various individual items on it. For demonstration, represented with a traditional size of 8x8 and with six various items on it (basic colored shapes), with highlighting the next possible move by showing a transparent cube over it.
- Controls (input) by touching/mouse clicking on the selected item and draging it to the desired side. Pinch-to-zoom or mouse wheel for zooming the view.
- Very simple GUI with a "New Game" button and a score label.

The second iteration proposed a couple of new features:

- Simple drop/swap/destroy/highlight animations.
- Ability to destroy items in a given shape, for example destroying objects in a cross shape (n vertical and horizontal from a chosen point, n=3 → 12 in total). The chosen point is selected randomly as a "bonus" (represented by a darker color, triggered if matched).

*Module code: SE3U608, University of South Wales, Faculty of Computing, Engineering and Science.

†E-mail: 13041290@students.southwales.ac.uk (Brno University of Technology – Erasmus: xkimer00@stud.fit.vutbr.cz).

[1]Autumn term 2013/2014.

[2]CEMAS – Centre of Excellence in Mobile Applications and Services, http://cemas.mobi.

[3]E-mail: andrew.ware@southwales.ac.uk.

[4]E-mail: dann.rees@southwales.ac.uk.

3 Implementation

The first iteration used Unity 4.2, but the final framework is implemented in the new version, Unity 4.3 (projects are not backward compatibile).

There are four main asset types used in this Unity project – scenes, scripts, materials and prefabs. The main game scene consists only of a GameController object (empty GameObject with the GameController script attached), main camera and a directional light. Scripts, which are the core of the implementation, together with materials and prefabs, are described in the following subsections.

3.1 C# Scripts

The game logic is implemented across a couple of C# scripts (.cs files):

Match3 implements the main game logic without dependencies on Unity (class does not inherit from MonoBehavior). It contains public methods for making valid new board, swap checks, getting arrays of matches, drop swaps, destroyed items and next possible swaps, and generating new items instead of destroyed ones. It also implements the "bonus" feature. The method for determining next possible swaps based on patterns is inspired by the Rosenzweig's (2011, pp. 297-300) approach.

GameController is the main game object for controlling the game in Unity. It processes the inputs, communicates with the game logic script and updates the game objects with animations according to game state. It also creates a simple GUI. The script has a public interface for adjusting the game settings from editor – e.g. board size, game objects representing board items, and other.

BoardItem is a script for adding the animation feature to the items on the board (it is attached to every game object representing the item on the board). It also contains some helper properties, like its coordinates on the board (e.g. for picking).

ScalePingPong is a very simple behavior which performs a ping-pong scaling of an attached object. Used for the highlight animation of the next possible move.

CameraZoomPinch performs a pinch-to-zoom (two fingers) camera control (with a mouse wheel support as well). The script is attached to the main camera and changes only the field of view (FOV). Adapted from `http://answers.unity3d.com/questions/63909/pinch-zoom-camera.html`.

GUIScaler automatically scales GUI elements on high DPI devices like mobile phones. Just copied from `https://gist.github.com/darktable/2018687`.

The more detailed descriptions can be found inside the individual commented source files.

3.2 Materials and Prefabs

The materials and prefabs used in this work are very simple – only for gameplay demonstration purposes.

The materials use simple Transparent/Diffuse and Trasparent/VertexLit built-in shaders with basic colors. The transparent shaders are used because of the destroy animation – item disappears by setting alpha channel to zero. The basic color is adjusted in-game as well – for highlighting the color of selected item (it is multiplied by 1.5), or darkening the bonus' color (multiplied by 0.5).

The prefabs are used for representing items on the board – simple cubes and spheres with attached box collider (for input – picking), material and BoardItem script (both discussed above).

4 Results

The final version of developed framework running on the Samsung Galaxy S4 (Android) is shown on the figure 1. Figures 2 and 3 show the project in the Unity Editor. The results can be downloaded from the internet:

- Gameplay video: `https://www.dropbox.com/s/uagfwp0tjrqariu/Match3_v02_gameplay.mp4`
- Unity 4.3 project: `https://dl.dropboxusercontent.com/u/17436248/Match3/Match3_v02.zip`
- Android debug apk: `https://dl.dropboxusercontent.com/u/17436248/Match3/Match3_v02_debug.apk`

Note – the first iteration of the game (without animations and bonuses; in Unity 4.2) can be obtained by replacing the version number in the hrefs with a previous one (`v02` → `v01`; except for the gameplay video).

5 Conclusion

The playable Match 3 game framework with animations and other features has been developed according the instructions. This report covered the design, implementation and results of the project.

The future versions could contain new animations/effects support, bonuses, time constraints, better graphics, and many more following the design document – this is just a beginning of development of the final product.

References

Rosenzweig, G. (2011) *ActionScript 3.0 Game Programming University.* 2nd edn. Indianapolis: Que Publishing.

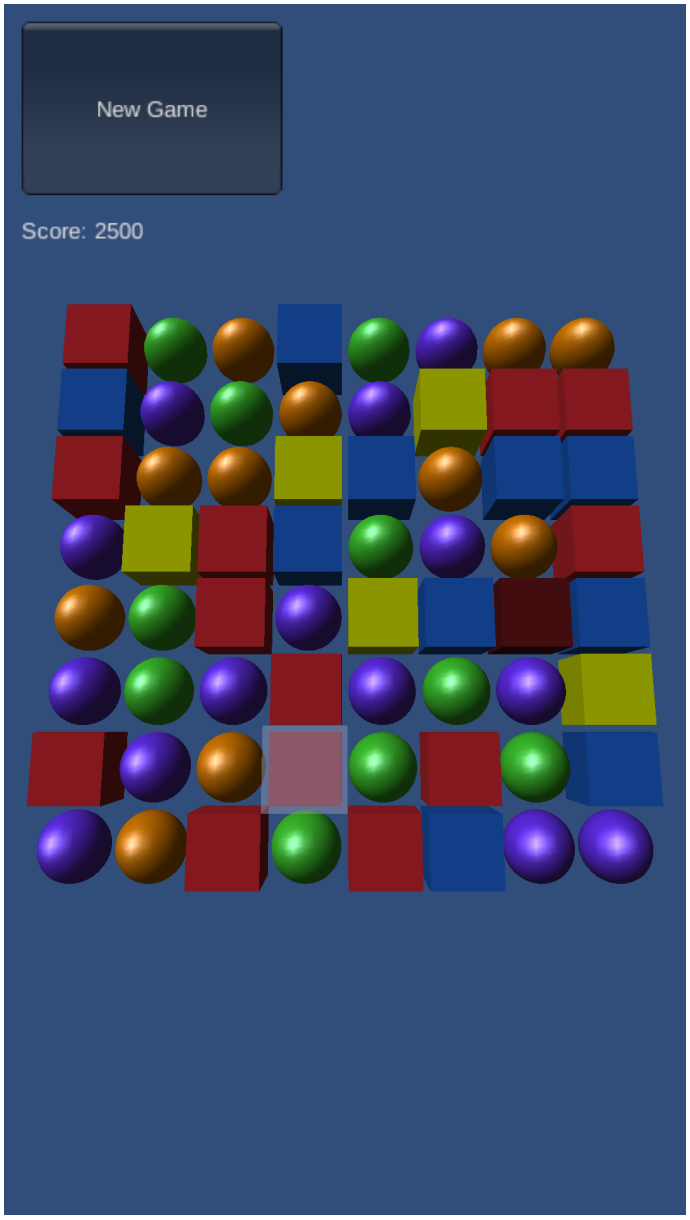

Figure 1: The game running on the Samsung Galaxy S4 (Android 4.3).

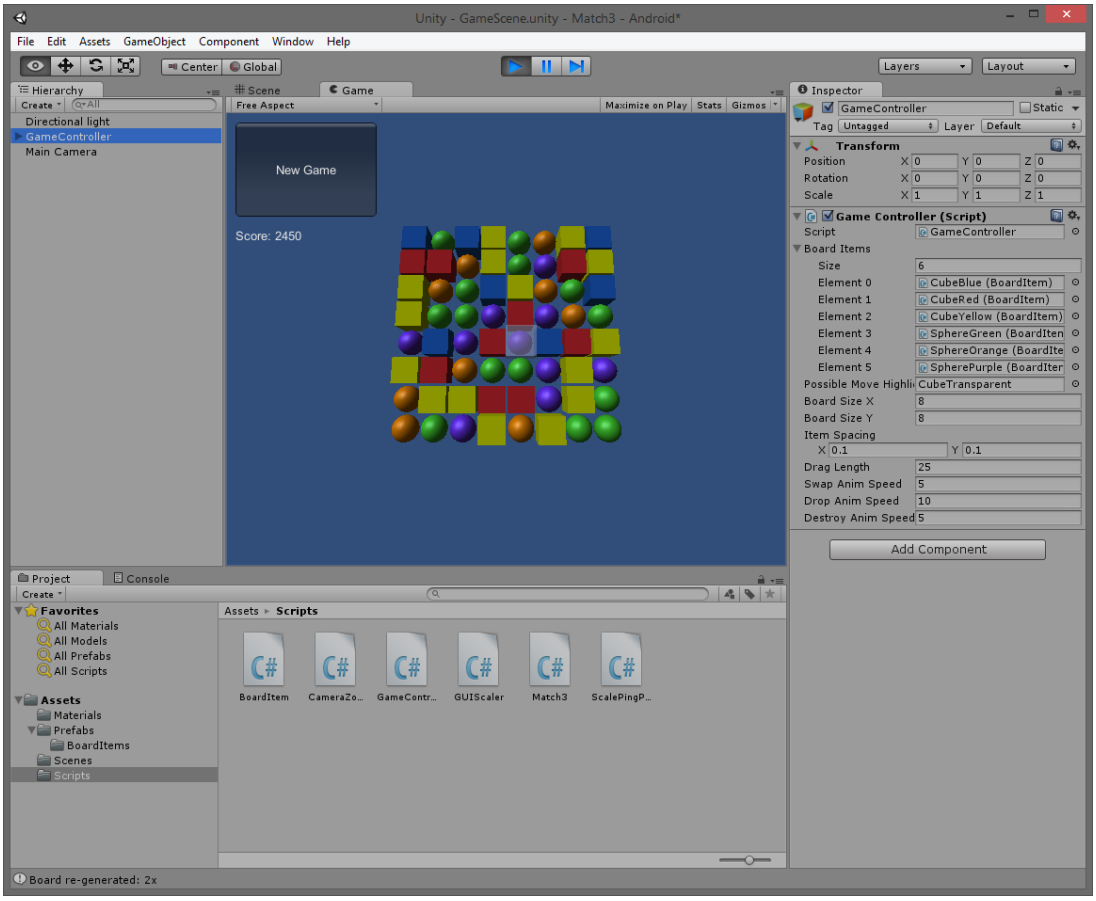

Figure 2: The game running inside the Unity Editor. The GameController interface is shown in the Inspector.

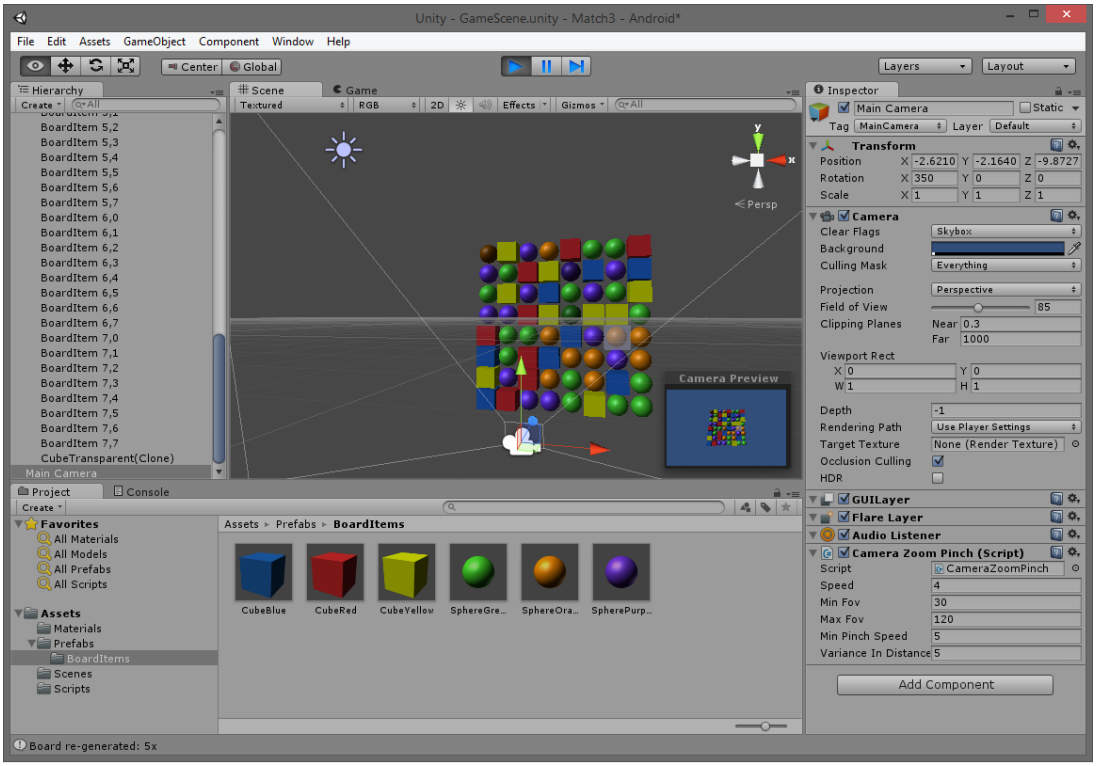

Figure 3: The game scene in the Unity Editor (while the game in running).