Development of 2D Game with Construct 2

Antonio Šag and Tihomir Orehovalčki
Faculty of Informatics, Juraj Dobrila University of Pula, Croatia
{antonio.sag, tihomir.orehovalcik}@unipu.hr

Abstract - Developing a computer game presents a big challenge, since creating a single product requires much more knowledge and experience in different areas than just programming. However, the question is how to bring your work to life in hours and days instead of weeks and months. Using the Construct 2 drag and drop method, we are able to implement visual materials into our game, and the principle of programming is reduced to the events option, which helps us to assign actions through selected graphics icons. Application instant preview allows fast prototyping and iterative development which enables testing during the entire creation process, making it much easier to detect and resolve problems. It gave us flexibility and the ability to see how the prototype of each version actually responds to certain actions, conditions, and events. As a final point, with a good organization and understanding of powerful event system, flexible behaviours, stunning visual effects and capabilities provided by the Construct application package, it is possible to do any conceivable 2D project.

Keywords - Game Development, Construct 2, Game Optimization, GUI, Interactive Review, 2D Project, Procedural Mode

I. INTRODUCTION

In the 21st century, computer games are one of the most popular forms of entertainment. Games have entered into every aspect of everyday life and they do not have the age limitations. The development of this form of entertainment is very demanding because the multimedia shape includes integration and adaptation of videos, animation, 3D, text, audio and motionless picture. The development of multimedia computer games is a very complex job which requires the team of experts who have the understanding of theory and practical methods used during the development and who know how to use professional tools for the development of the multimedia games. The motivation for the game creation came from the detailed research and analysis of the so-called “old-school” 2D games which marked many childhoods. Throughout this paper, we will discuss the production of 2D platform game intended for one player. The name of the game is “Funky Student”, and the groundwork of production is based on the usage of Construct 2 architecture. The particularity of Construct 2 environment is in its adaptability to various platforms, such as browsers that support HTML5 and operational systems like Linux and Windows as well as the interactive compilation of the game at any moment, which enables realistic overview of the game functionality to the programmer. Graphic user interface is conceived with the principle of adventure platform game „Super Mario”.

The remainder of the paper is structured as follows. The second section will briefly introduce Construct 2, present comparison and features of the competitive 2D game engines and explain which games motivated us to start our own project in that respect. The essential part of the paper is the third section in which the whole process of game development is thoroughly explained. Conclusions are drawn in the last section.

II. BACKGROUND TO OUR WORK

A. Construct 2 environment

Construct 2 is a very simple commercial application for the creation and design of two-dimension games developed in early 2011. In Construct 2, the classic coding is replaced with dragging and dropping game objects. Games developed in Construct 2 are implemented in HTML5 and Javascript languages. Several commercial computer games were developed with Construct 2, some of which are available on Steam [2] such as Airscape: The Fall of Gravity, and The Next Penelope [1].

B. Comparison with the competitive 2D game engines

Adobe product called Adobe Flash Pro (Adobe Animate) is the industry’s leading animation toolset which lets us create apps, ads and amazing multimedia content that moves across any screen. It allows us to draw art assets in the engine using vector art and also animate it in-engine using key frames which is very convenient and simple to use. Programming the animations is relatively easy as well. Flash uses the Action Script 3 programming language which is similar in syntax to JavaScript or C#. Flash is still widely used in game development these days. It has a free 30-day trial option, but once the trial is finished we must buy a license, unlike Construct 2, where we can work with a free version as long as we like.

Game Maker Studio was originally released by Yo Yo games in 1999 under the name Animo. It was originally made for creating 2D animations but quickly moved on to being a very robust 2D game development tool. Overall this engine is known as an introduction to game dev tool or a hobbyist tool. Games made in Game Maker Studio are generally made using pre-defined events. Example of pre-defined event will be: "when we select create new event defined action like “Move Left” [7]. This means when the Left is pressed the assigned character moves left. This makes it easy for people to make simple games without knowing how to program. The following
Figure 1. summarises the features of each of the tools covered and provides a simple comparison of the tools.

C. Current similar games

The idea of the game is generally perceived as a so-called platform way of playing where the user controls the main character and the levels represent platforms which contain various objects such as enemies, key to unlock the “door” in order to pass to the next level and alike. The basic approach to the 2D platform game is easily related with a well-known game “Super Mario”. Also, the principles, the interface and the access to other 2D games contributed to the plan itself and the development of the game. On that list are games such as Icy Tower, Flappy Bird and Angry Birds, which have represented the “addictive era” of 2D games with its simplicity and adjustable user interface.

III. Process of developing the 2D game

Before the actual development of the game, during the pre-production part, it was necessary to come up with the concept of the game itself and to design its components. The game itself is conceived so that the main player is located in the 2D platform where its goal is to collect three signatures in order to successfully finish the level and open the door of the university to pass to the next level. The task is made more challenging because of the different enemies which are located on various positions in levels. Moreover, there are gems on the platforms which serve as guides through the level. The player has the power we can relate with the popular game “Super Mario” where he can destroy the object of the enemy with his jump, or more precisely, the fall. The game has three levels accompanied with special sounds which accentuate particular action and with the background music which gives the additional charm.

A. The look of the layout

Layouts represent levels in the game. The reference of the motifs in the background are characteristics related to the city of Pula as shown in Figure 1. The second and third layout were generated according to a conceived initiative. All the content and objects we enter into an active scene will automatically be connected to the active layout.

B. Layers

Layouts consist of layers which enable us to classify the imported materials in groups. When importing materials and objects, it is important to do the process of sorting by the layers as shown in Figure 3. This process is done in specified window for layers thus contributing to the easier workflow, better organisation and visibility of the project itself.

C. Event sheet and events

Every layout has its event sheet in which all the steps related to the addition of the events, addition of actions and other events as depicted in Figure 4. Events are the main characteristics of the Construct 2 that facilitate game development. They eliminate the need for complicated scripting in programming languages with fixed syntax and overcoming obstacles when developing a game by careful studying the enormous quantity of content in help, tutorials, and guides. To understand what is the best way to solve a certain problem, Construct 2 offers us the programming with the help of logical blocks of the system. In other words, events are collective blocks. The basic concept of
events is that conditions filter the cases which fulfil the state, and then the actions are done only for those cases.

D. Behaviours

Behaviours give predefined functionality to object types. Construct 2 includes 26 behaviours, some of which are shown in Figure 5. One example of behaviour can be “solid behaviour” which makes other behaviours react to the object as if it were an impassable.

E. Effects

Effects change the visual appearance of the object. They can be added in the Effects dialog box. Effects can be added to the layers and layouts, although the effects which are blended to the background cannot be used in the layout. They are sometimes categorised as shaders or WebGL shaders, only because that technology enables the appearance of those effects. Construct 2 has over 70 different effects which can be used in its library, and some of them are shown in Figure 6.

F. Families

In Construct 2, families are groups of objects. All kinds of objects in the family have to be from the same plug-in (for example all Sprite objects and not the combination Sprite and Tiled Background objects). Families help us to avoid the repetition of events, as depicted in Figure 7.

G. Plug-ins

Plug-ins determine the type of objects. Construct 2 contains a specific window for the plugin list. Some of the plugins are shown in the Figure 8. For example, Sprite is one type of the objects while video, keyboard, and button are completely different types of objects.

There are three main kinds of plug-ins:

- Visual plug-ins (e.g. Sprite) occur in appearance and they draw something on the screen.
- Hidden plug-ins (e.g. Array) are set on certain layouts, but are not drawn on the screen.
Project plug-ins (e.g. mouse, audio) are added to the whole project and can be added only once.

Data & Storage

- Array
- Dictionary
- Local storage
- XML

Form controls

- Button
- File chooser
- List
- Progress bar
- Slider bar

General

- Sprite
- Particles
- Shadow Light
- Sprite
- Sprite font

Figure 8. Plug-ins in Construct 2 environment

H. Objects

Object is the representation of certain idea. Mentally, when you imagine a certain idea, it can be something like a book or a ball. Although you know very well how a book looks like, you also know there are many unique books. Every instance of any object in Construct 2 assigns a unique value known as UID (Unique ID) [1]. As a visual representation of objects inside of the game, all the objects (and one animation of each) are shown in Figure 9.

Figure 9. Objects of the “Funky Student” game

I. Object types

One of the main elements of the game design in Construct 2 are object types. They define the class of some object. For example, we have types of objects “Player” and “Enemy” which come from the same (Sprite) plug-in. They have different animation and will behave differently with the help of events although they come from Sprite object.

J. Object types instances

Object types instances allow to each object (e.g. enemy) to save its own life value. In short, the variable is a value that can be changed (or vary) and is stored separately for each instance.

K. Main character

It is necessary to specify in advance what a certain object will be used for. Since we talk about the main character, it is necessary to specify in advance its movements, namely rig animations, which prove to be very helpful in animation of movements, assignment of events and player action, as presented in Figure 10. It is a drag and drop process of the set of animation pictures in the Sprite object. In that manner, we create the initial animation in which we need to specify collision on each of imported pictures. There are five animations for the main character: walk, stand, jump, fall and dead. In order to be able to move through the platforms, it is necessary to assign “jump and run” style of movements to the player, which is done with behaviour Platform, that enables the object to move left, right and the possibility to jump.

Figure 10. The main character animation

L. Enemies

By employing events, we can control movement of enemies through the work space independently of the main character or adjust their movements depending on main character. The process of creating the enemy is equal to the one creating the main character. Figure 11 shows events for simple automatic movement of the enemy that will during the realisation of the Solid behaviour, in this case the wall, do the rotation for 180 or -180 degrees on the x axis and keep moving.
1. Destruction of enemies

It is common to all the enemies that in the case of “TRUE” collision with the main character (player) they are destroyed, as shown on an event tree in Figure 11. During the “FALSE” collision, one life is taken away from the player and the Flash behaviour is initiated which gives the object the possibility to flash in a way that it rapidly turns on and off the visibility of that object. Events for “FALSE” collision are shown in Figure 12. Finally, the sound record is initiated which demonstrates the collision, or rather “FALSE” collision with the enemy. In order to destroy the enemy, the player must jump and fall on the enemy to do the destruction of the enemy object.

2. Global variables

Global variables store their values between layouts. Events from any layout can access to any global variable, even if it was created on the other event sheet which is not turned on.

In this project there are parts which follow each other and change in time. Those are global variables related to turning on and off the background music and special effects, as well as the variables connected to relations, namely the state of the game (e.g. win or lose) and numerical variables which refer to lives of players and the sum of gems through levels. All stated global variables inside the game are presented in Figure 14.

3. HUD

The word HUD (head-up display) represents the insight into the player’s progress, things (s)he currently possesses (e.g. lives) and things (s)he collects (e.g. signatures and gems) as shown in Figure 13. It is necessary to create a special layer for HUD in order to fix and keep its position on every level. It is most commonly created from textual-numerical object connected to some variable. In this project, HUD is used so that the player can get insight in his/hers lives, gems and signatures.

4. Passing the level

The ultimate goal of the player is to finish the game, which is achieved by passing from one level to another. Each level is specific and has different structure of platforms and enemies. As presented in event sheet that is depicted in Figure 14, the process of passing the level is as follows: when the player collects three signatures, the system will start the action of changing the animation on the “university door” and the colour of the flag from red to green which will result in visual simulation of the door opening.
Q. Main menu

The last phase in the game development is design of the main menu. As shown in Figure 17, the main menu is here to present the game home screen to the player and introduce him/her the basic controls. During any moment of the gameplay, main menu can be reached by pressing the escape (ESC) button or by clicking on the home button.

R. Business plan, target group and game distribution

The first step is to identify and understand the target group. During the aforementioned, it is necessary to take into the consideration the following factors: geographical location, economic factors, and age of target players [4].

The game “Funky Student” is aimed at younger users (but not necessarily) and is based on the free to play method (F2P), where the game would basically be free, but it would have monetization options. The price of the game and the price of additional features in the game should be maximally optimised with the consumer boundaries of the target group. A price which is too low or too high will directly influence the monetization potential of the game. It is also necessary to investigate all the available options of distribution platforms for the game. The most common way of distribution of the multimedia computer game is the digital one.

Digital distribution of not only accessories, but also the entire multimedia computer games has become the most common way of buying games. Online distributors usually take around 30% of income, and if they take less, it usually means lesser penetration to the market. The choice of the platform and the canal of distribution will play an important role in the overall incomes from the game. Services such as Steam with impressive base of 35 million active players has as much as 80% of digital distribution of games on PC platform according to estimations for 2018. Since nearly any game can be published on that service, it is ideal for small development teams who would never have their game published by the traditional publishers. It can be said that Steam lead to the increase of Indie games which do not require big investments in the development and teams of just several people are enough for their making [6].

IV. Conclusion

The process of making a computer game is a complex one. Depending on the type of the game, it is necessary to choose the right event hierarchy, namely the processes which need to be followed and maximally optimised to decrease the influence of bugs and unwanted actions. Drawing on old-school 2D games, objects (main character, enemies, platforms, gems, signatures) have been created whose actions (events) are added in the Construct 2 environment. When the user successfully completes a particular level, the university doors open. For this part of the game, the reference is made with the action of exam taking. Construct 2 introduces the game designer with the making of user interface in interesting and dynamic manner. Events govern the objects which are means the game backbone is made from. The environment is accessible and easy to use, with an interface which is susceptible to various expansions/accessories from the third party plug-ins as well as plug-ins programmed in JavaScript. By importing prepared objects, one gets a template of the game with three levels and user interface. In order to achieve that game looks good and photo-realistic and at the same time to have realistic simulations, one would usually need to invest a lot of time, effort, and have expertise in various fields and environments. Construct 2 enables anyone to create a game in easy and straightforward fashion. The aforementioned feature of Construct 2 attracts the interest of many programmers, especially the younger generations who are just discovering the world of programming. In that respect, the Construct 2 environment would be an excellent representative in the world of interesting and instructive ways of programming.

REFERENCES