Game Design & Development

WS 17/18

Post Mortem

Santas Christmas Run

by Kevin Kuntz, Alexander Huber and Christopher Biehl
Saarbrücken, January 31, 2018.

Contents

1 Game Design

Our Game Design Document described a 2D sidescroller game with a procedurally generated map. Further it described how the player can interact with the environment.

We have a camera that is moving forward as soon as the player starts moving. After that the camera keeps moving even if the player stops. Inside the camera-sight the player is able to move forward or backward. This gives him the ability to calculate which platform he can reach next.

Our player can move through the world by using the keys A, D (left and right movement) and space (jumping). He has 5 life points which are indicated by heart-icons in the hud. His goal is to achieve as much score points as possible.

There are two types of collectibles. First we have cookies that restore 1 life point each. Second there are collectibles that affect your score points. These are indicated by candy canes (two different sprites).

To make the game more difficult we implemented a few mechanisms. The first mechanism is the increase of speed. Every 2000 score points the overall speed will be increased.

Different platforms is a second mechanism. We implemented our map generator so that we could spawn more than one type of platforms. The algorithm will spawn neutral platforms and dangerous platforms. Neutral platforms will not harm the player in any way, but dangerous platforms will affect the player in some way. Dangerous platforms are lava and ice platforms. Lava will decrease the players life points by 1 if he touches them. If he keeps staying on lava his life will be decreased more. Ice platforms do not affect your life points but they will accelerate you. A good aspect of this is that the player can make use of ice platforms to cover a greater distance with a single jump.

The third mechanism is throwing snowballs in the players direction. If you get hit by a snowball your life points will be decreased by 1 and you will be pushed back a little bit. Every 500 score points there is a chance that one or more snowballs are thrown towards the player. This way we can make it more difficult to stay on the far right side of the screen.

As described in our Game Design Document we implemented an algorithm to generate our map. It will spawn enough platforms in advance so that you do not see the generation process while you are playing.

2 Tasks of the Team Members

Our team members were Kevin Kuntz, Alexander Huber and Christopher Biehl.

Kevin Kuntz together with Christopher Biehl mainly worked on our map generation algorithm, platform effects (decreasing health, acceleration), player (including movement), collectible-system and sprites we needed for this. Kevin also worked on the parallax background.

Alexander Huber first worked on HUD and GUI, then implemented the snowball mechanism.

3 Used Tools

As our development tool we used the Unity Engine. Our source version control tool was Git with a GitHub-repository. We also used the GitHub project planning functionality to manage milestones and issues.

We used GIMP (GNU Image Manipulation Program) for our sprites, because we decided that we want to create our own sprites and not to use sprites from the asset store. Creating our first sprites was not as easy as we thought, but after some time we figured out how to use GIMP. An essential part was to understand the working of layers in gimp. The design and creation of sprites consumed more time than we thought, but it is good to see that the game is completely build from our work. Creating sprites also gave us some variety of work.

We also used Google Drive (Google Docs etc.) really often. It gave us the possibility to real-time edit documents with multiple users.

4 What Worked Well

4.1 Realistic goals

We think that in general we were able to focus on the main things and kept our goals simple and realistic.

4.2 Working with Unity

Unity itself is a great tool for developing games. The internet provides you with lots of tutorials and examples. The documentation has a good structure and is well explained. Unity is feature rich and after getting familiar with his structure and components the development worked well.

4.3 Code-freeze

We set the code freeze to the 26.02.2018 because after the first two presentations we realized how much work we have to afford for the deliverables. They needed more work than we expected at the beginning. So we decided that we need at least one whole week to made a good last presentation and create good documents.

We finished all features one day earlier. After that we just needed to do some bug fixes and adjusting parameters like speed, gravity, spawnrate etc.

5 What Worked Not so Well

5.1 GitHub Issues

We should have used the issue tracking from GitHub earlier and more frequently. It helps a lot to structure your development cycle. You get a good overview of all assigned tasks. And you can always keep track on which task the other team members are working on.

5.2 Unity and Git

Unity gives you the possibility to use an own version control. Now after the development cycle we came to the conclusion, that we should have used the version control from Unity itself. Unity produces a lot of overhead of metadata. Unluckily these metadata are binary files. So git has lot of trouble to compare and merge them. We often needed almost an hour to remove all conflicting metadata, which was really annoying.

5.3 Architecture

We started too late with our main implementation. Reasons for that are other courses and a wrong architecture. We did not make enough thoughts about our architecture. It was our first game we implemented and thought, we could just implement the game.

In the middle of our project we came to the conclusion that we need to rethink and change the architecture. That was difficult, because we already had some main features and mechanics implemented like platform generation and movement.

We decided to create a main game manager, which is our highest instance. It can provide every class with all values they need and also keeps every class decoupled from other classes. With the new architecture we had no problems to implement new features.

6 Summary

We already were interested in game development before we participated this course but never really tried it out seriously. Participating in this course really helped us getting somewhat familiar with Unity and the game development process in general.

The lectures were really helpful to get an insight into game development process and all the other things that come with it like creating a design document, and also a game concept in general. We never thought of these things before.

It was also good to see how the other teams were working and which engines and tools they used.