Santa’s Christmas Run

Game Design Document

TEAM_CK

Kevin Kuntz, Alexander Huber, Christopher Biehl

Game Design & Development
WS 17-18

Table of contents

1 Game outline

1.1 Concept

The player is playing a character that moves through a 2D map from left to right. He is going to earn points over time as well as by collecting items and killing enemies.

The main goal is to achieve a highscore and to beat the previous highscores.

He might encounter different difficulties like enemies or obstacles. He can avoid them or fight them to stay alive.

Our main focus is the map generation. We want to create a map generator that procedurally generates the map as you play the game.

1.2 Story

It is the 24th of December. Santa Claus has delivered all presents to the children. After a long night he comes home to the north pole.

Just when he sat down to relax and enjoy the remaining christmas time with his wife, one of his elves rushes to him and tells him that something horrible happened: While the children are sleeping all the presents have disappeared!

Santa now has to bring back all the presents to save the children's christmas.

1.3 Sprites

Of course we want to have some sprites that match our story description. Because of this we are going to draw some easy christmas-influenced sprites on our own. If we already have some sprites we are going to show them at the chapter of each game object.

1.3.1 Drawing the sprites

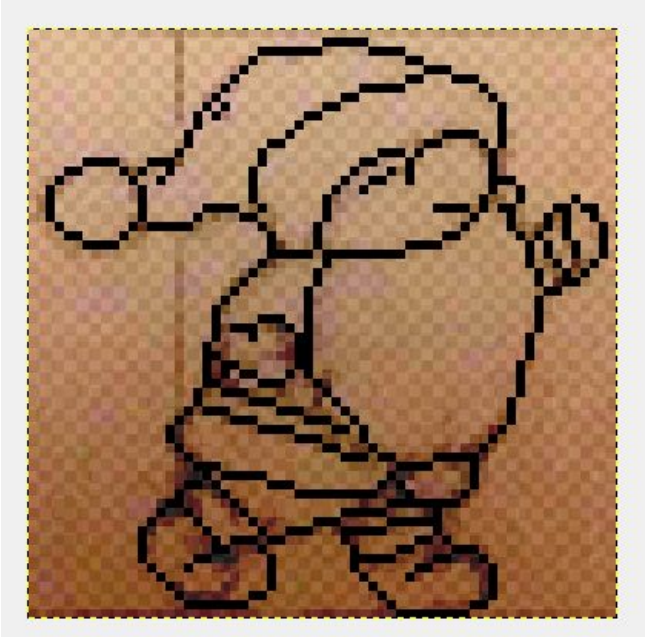
First we think of an item or character we need in our game. This could be for example a candy cane or the santa claus.

Then we draw or look up a sample of this character.

We open up GIMP (GNU Image Manipulation Program) and create a new 128x128 image.

Then we load this image, copy it, paste it into our empty image and create a new empty layer. Then we reduce opacity of our sample image so that we can see what we are going to draw.

From now on we just have to paint all the outline pixels in a dark color, and then fill the inner parts with a color we like.



And now fill everything with fitting colors.

Now we just turn off the layer with our sample and we have our little sprite for our santa.

2 Character

The player will play the main character, which is Santa Claus.

2.1 Look

We have drawn a first version of our Santa Claus-character. He looks like a classical Santa Claus you might know from movies.

The sprite for our santa character.

3 Gameplay

3.1 Genre

The game is going to be an endless runner: this means it is a 2D Jump'n'Run with a procedurally generated map.

3.2 Player mechanics

The player starts with two life points which are represented through candy canes for example.

3.2.1 Movement

The character is controlled with w,a,d and space.

- W/up arrow - jumping
- A/left arrow - moving to the left
- D/right arrow - moving to the right

If 'W' is pressed once the character will do a normal jump, but when 'W' is hold it jumps higher.

3.3 Difficulty

We want to increase the difficulty over time. This means based on how high your score is, there is something going to happen that increases difficulty for the player.

3.3.1 Speed

The speed will increase based on how long you are alive.

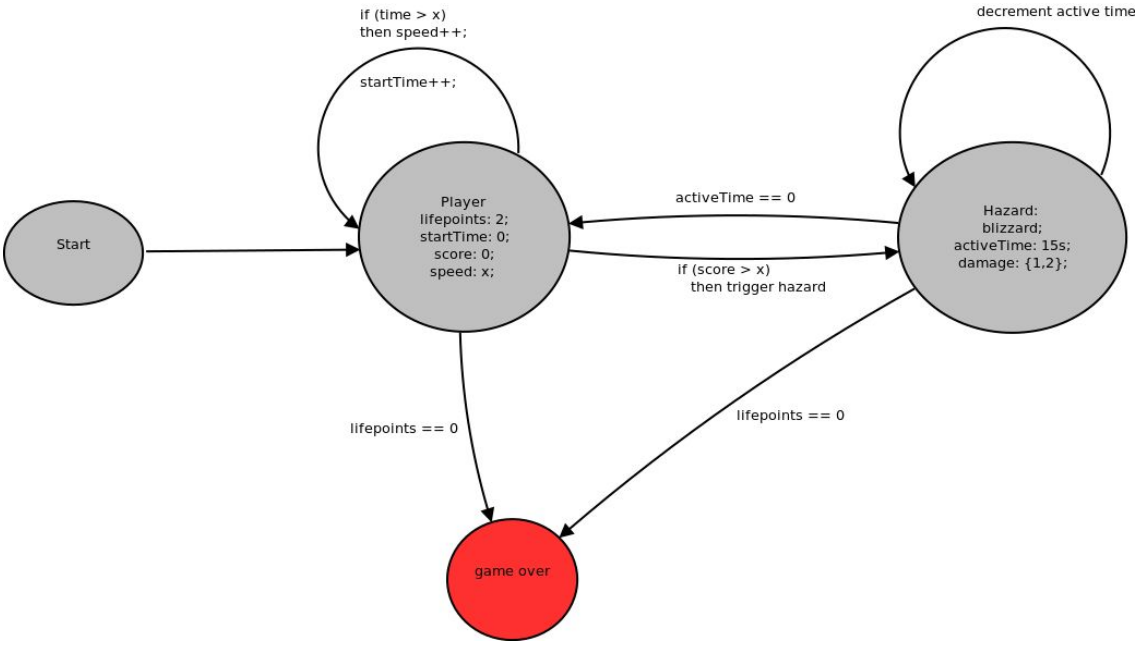
3.3.2 Non-intelligent obstacles

If you reach a certain score a blizzard with flying objects will appear. To survive, the player has to dodge those objects. There will be also some classical obstacles like big gaps to jump over or ground that does damage.

3.3.3 Enemies

If map generation works out well, we may be adding some kind of enemy. Limitation will be that they won't have a complex intelligence. They will just be like the koopas (turtles) from super mario.

3.4 Flow



This finite state machine shows the flow of the game. The player spawns in the beginning of the map with the following attributes:

- 2 life points
- initial movement-speed
- initial time (0)
- initial score of 0

The speed will we increased determined by how much time has elapsed since the start.

```
if (time mod 20 == 0){
     speed++;
}
```

As soon as the player reaches a certain score this will trigger an event like a blizzard. The player needs to avoid the flying objects to survive.

```
if (score mod x == 0) {
     startBlizzard();
}
```

The player is game over when his life points reach zero.

4 Game world

4.1 Settings

At first the game will have one setting, which will be a snowy landscape with (christmas) trees and some clouds.

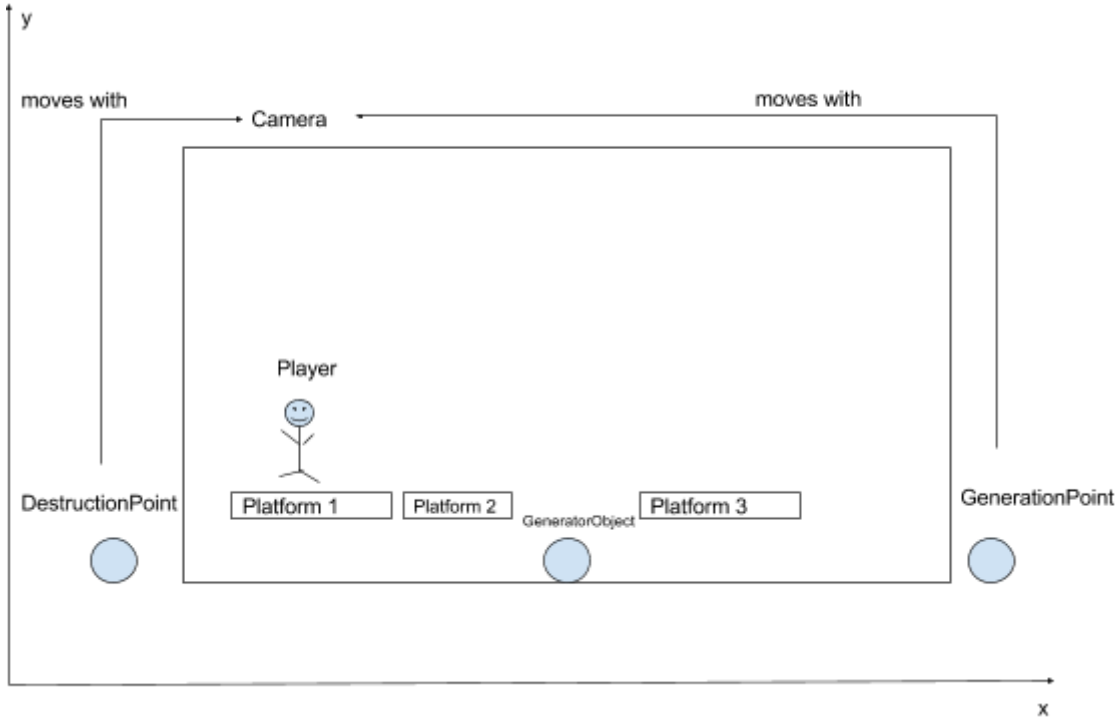
Later on, depending on how map generation works out, we may be adding some other map-settings.

4.2 Procedural generation

As mentioned in the concept chapter, our main focus will be map generation.

Our goal is to create a generation algorithm that generates platforms as you play the game.

The following part contains a first idea on how this algorithm may look like.



We have one area which is our camera (the rectangle) and two points that are attached to our camera. Whenever our camera moves, our DestructionPoint and GenerationPoint are moved also. Then we have our GeneratorObject (the object that has our algorithm attached). This object is not moving when the camera moves. Instead in every frame we check whether the x-coordinate of our GeneratorObject is less than the x-coordinate of our GenerationPoint. If yes, then we move the GeneratorObject to the right by a random amount between x and z and also spawn a new platform.
If the GeneratorObject is further to the right than our GenerationPoint we just have to wait until it's "behind" our GenerationPoint again.

Of course this is only a first idea that needs improvement especially if we want to have platforms that are not limited to 1 unit in height.

5 Future: Monetization

There are lots of possibilities to add monetization to a game. For this Jump'n'Run type of game we think the best possibilities may be the following:

- translating score points into an in-game currency
- offer new characters/map-settings (sprites) for in-game currency or real money
- offer boosters for in-game currency or real money
- offer in-game currency for real money