02823

Computer Game Prototyping

Circus For A Psycho

Date: 15 December 2014

Eleftherios Manousakis
Student Number: s141714

Jérémy Rombourg
Student Number: s142279

Sigurd Knarhøi Johannsen
Student Number: s042910

Course responsible:

Michael Rose

Contents

1. Introduction

The game is called **Circus for a Psycho** and is a 2D side-scrolling platformer. Technically it is a take on the NES game Super Mario Bros. [1]; there is basically no plot. The player runs, jumps, swims, collects bacon and reach a goal at the end of the level. Instead of Super Mario's jumping on enemies, we have a shooting element - like in the NES game Metroid [2] and we've implemented an additional merciless point system (the balloons) that reminds of collecting extra lives in the SNES game Donkey Kong Country [3]. The main character design is inspired from an online comic called Cyanide and Happiness.

2. Detailed description of the game

a) Game rules

The player must traverse platform levels of increasing difficulty. Also a better score is awarded for a more daring performance and for a fast completion. The game is over after beating three levels and defeating the evil circus director. The player dies, if he takes too much damage from projectiles, fire and spikes or if he falls into an abyss.

If the player reaches a checkpoint or finishes a level, the progress is saved and earns the right to restart from that point.

b) Controls

The controls were meant to be simple, intuitive and relaxed. Running and jumping comes natural to 2D platformers. This is an advantage to us, as designers, since we won't have to waste the players' time explaining much about the basics. To add to a relaxed control, we decided not to put a time limit on the levels and to give the player unlimited lives, so the player could explore the controls in his own tempo.

To avoid making the game too simple however, we decided to add a shooting element and interaction, in the form of carrying an object from one location to another.

The player move left and right using 'A' and 'D', jumps on 'space' or 'W', throws knives with 'left mouse button' and picks up/throws away hay using 'E'. 'Esc' activates and deactivates the pause menu.

c) GUI

The GUI in our game is created using Unity's old GUI system. From the start menu the player can start or quit the game. Within the game the player's score is presented in the upper left corner. Just below are a counter for bacon and a counter for Balloons. If the player presses 'Esc' during gameplay, a pause menu will appear, allowing the player to return to the game ('Esc'), return to the main menu, restart the level, un/mute the sound or quit the game.

d) Collectibles

The bacon serves to give the player a reason to take more chances and in general to attempt dangerous manoeuvres. It also gives the player a purpose, as in wanting to try and collect them all, which in turn also provide for added re-playability.

The balloons may at first look like “just another bacon”. However as the balloons are slowly flying upwards and are never reset – their true purpose is revealed as adding a disguised time trial to each level. To be able to catch every balloon, the player is actually required to beat the level in a certain fast tempo. To further distinguish balloons from bacon, we decided that the player had to collect the balloons, by popping them, using throwing knives.

Health crosses provide the player with renewed health. These especially come in handy when facing enemies later in the game.

In level 2 we introduce collecting hay. Jumping into a haystack, the player can take along some hay, using the action button 'E'. While carrying hay, the giraffe will follow you.

e) Other features

Spikes are a well-known feature of platformers and require little to no introduction. The player takes damage on contact and is pushed back slightly. A different version of spikes surprises the player by popping in and out of the ground. These will kill the player instantly.

Clouds and water bubbles are created as particle emitters and their function is to add volume to the 2D levels.

Stars in the background in level 2 and 3 are meant to hint the direction through the level.

Fire and smoke are created from a collection of particle emitters and a point light source.

Throwing knives is the player’s weapon of choice, but are also used by some of the circus enemies. They fly in a straight path.

The camera will follow the player around, using interpolated regular expression. Also it gives the player a small margin, before following.

The player and other characters are 2D sprites, to which we apply animation, via animators. The character sprites are flipped, by scaling -1 in the x-direction to fake the character turning left or right.

The main character also has animation for running, jumping and falling. Of which the last two are in fact the same animation that is used depending on the velocity in the y-direction.

Fading is also incorporated between levels for a smother advancement feeling.

f) Volume space

The water volume changes the player controls and thus the entire game experience. The player needs to adjust to new controls - new rules. As long as the player is in contact with water, the gravity experience is completely changed and the player is allowed to jump whenever he desires. But water is also treacherous – the unaware player will sink to his death. To allow the player to get better acquainted with water, we added a safe water volume in level 1.

g) The platforms

We have six kinds of platforms: ground, trampoline, disappearing, jump-sensitive, moving and a giraffe.

The ground is the first platform the player experiences. It's static and can always be trusted. The player always knows where the ground is. It is the most basic, but also the least exciting and least challenging of all the platforms. Ground is mainly used to let the player catch his breath before another challenge appears. Level 1 primarily consists of ground, but occasionally introduces another platform.

The trampoline is the second platform the player comes across. It sends the player flying and is both a help and a menace. Having higher velocity can help the player reach higher ground, but also requires the player to steer during high velocity moments.

The disappearing platforms are cause of both relief and frustration, as its placement in the level dictates its function. In level 1, the platform may surprise, by helping a falling player, catching him before the abyss. The platform gives the player the extra amount of time needed to steer clear of danger. But when fighting a lion in level 2, the player is already aware of the platform location. The platforms are sending the signal of safe space, but in actuality, they can't be trusted through the entire fight. This gives the platform an entirely new purpose.

The jump-sensitive platforms appear or disappear whenever the player jumps. Encountering them late in level 1, the player needs to concentrate, but may do so in his own good time. It is first in level 3, when these platforms are combined with an incentive to hurry and irregular platform shapes, that the player himself is revealed as being his own worst enemy.

Lastly, the moving platform demand a certain pace from the player, but once you've reached the platform, you're safely transported to the next location. In level 3, another layer of challenge is added when these platforms are combined with the fact that the player can't move too slow or too fast, giving the player only a single or two shots at catching the platform, before it's too late.

The giraffe acts as both a platform and an AI, as the player can stand on its back and on its head. The giraffe is also an AI – it will follow a player that carries hay, and try to eat it, if you come too close.

The objective is to lure the giraffe to a location, where you can exploit the tall neck of the giraffe to reach a higher place.

h) Enemies and AI

We have three kinds of enemies; knife thrower, lion and circus director.

The knife thrower is the most common and the most forgiving enemy. He throws projectiles at a target in a given direction and if the player stands in the way, he will be sorry. The knife thrower is blindfolded and therefore stands still and pays no regard to the player's location. The knife thrower can, as all enemies, be defeated by throwing knives at them.

The lion is an optional boss to fight at the end of level two. The lion incorporates an AI, which effectively makes the lion a curious and very dangerous foe. The lion shoots fire at the player if he sees him and comes after him if he "hears" you. The lion will bite, if the player comes too close! Thus, the lion is an unrelenting foe and the best strategy may be to sneak past it.

The circus director is the evil villain, who makes it rain fire from the skies. He swoops down to kick you and there's nowhere to hide from him. Fighting the circus director requires timing and skill with the throwing knives. Killing the circus director, will make the player victorious in the game!

i) The levels

The three levels were designed to build on one another. The end of each level is denoted by a large particle effect, giving the player credit for a fast completion and transporting the player to the next level.

The first level is very laid back and introduces the basic controls such as running, jumping and swimming. You can traverse the level in your own pace and get to know the controls. There are no enemies. Level 1 introduces a number of features one at a time in a way so the player can easily restart nearby, should it go wrong.

Level two introduces moving platforms, enemies, AI and carrying objects. Moving platforms somewhat force you to play in a certain tempo and time your jumps. Enemies shoot at you and you have the option to destroy your enemies for more points. The player is required to solve a small puzzle, by carrying an object and thus luring a giraffe to a certain location, to be used as a platform. There's an optional boss at the end of the level.

Level three is the final test for the player, where he has to work his way vertically through the level. A rising fire forces the player to make quick decisions and take a lot of chances.

Reaching the end of the level you are for the first time forced to fight an enemy, to win.

j) Technical challenges

Moving Platforms

We had some issues with the moving platforms, as the character would not be translated along with them, when standing on top. We had to add the platform's velocity and direction to the character itself to solve it. Firstly, we tried parenting the platform to the player which caused further problems (e.g. when the player was below the platform). In our final implementation though, we are using raytracing to calculate the velocity of the platform and add it to the player. The result turned out to be much smoother in any direction.

Trampoline

In addition, we faced some issues with the trampoline. First, we tried creating a simple game object from which the player receives a velocity boost in the Y direction and plays an animation for the trampoline. In our first approach, we were only checking if the player is on the ground/trampoline to trigger the velocity boost collider. This gave the player an opportunity to jump *and* receive the boost, resulting in the player flying sky high. To solve the problem, we had to separate the trigger for the trampoline animation and the player's velocity boost. We managed to do this by separating the trampoline in four colliders. One collider triggered the boost, another triggered the animation and the two last were used as common ground, so the player could not trigger the trampoline from the sides or bottom.

Checkpoints

Checkpoints are used to save the character's progress for points and level advancement. Our first implementation included a script where the player had to grab the checkpoint, however in the later implementations, the player only has to pass a certain X or Y value to attain the checkpoint. With this approach we are giving each checkpoint a fixed number of pickups to be responsible for every time the player respawns, based on their X or Y value. This explains why the checkpoints objects are rings. This approach gave us also an advance in debugging the game, as we could choose from which checkpoint to start the game.

These three challenges took a lot of time and research to solve, but fortunately they led us to a better solution by leading us to the raytracing mechanism [5].

3. Implementation

a) Technical design

Controllers - defining the behaviour of NPC's, camera, player and abilities

CameraController

Camera script, which restricts the camera movement within the camera, bounds and follows, if needed, the player's position using lerp.

CharacterController2D

Handles gravity, collision and movement of the player using raytracing.

- Raytracing

Stumbling upon raytracing, we found out that there were several of our features that could be improved through this method. Basically, the character and NPCs shoot rays in front, backwards, below and also upwards. The rays originate from the center of the object and allow us to modify its movement, depending on what he stands on or if he is in contact with anything of interest. In this way, we can also calculate a different velocity for the player when walking on slopes.

- Water Volumes

We found a way of creating different physics volumes like water, to provide the player alternative physics movement parameters by implementing alternative control parameters whenever they were present and returning to the original control parameters otherwise. This is handled through trigger 2D colliders.

ControllerParameters2D

Contains the default parameters for jumping, velocity, gravity and slope limits. This approach was chosen to be able easily to override the parameters when in different volumes. This is a serializable script to be able to change public variables from the inspector, as this is not a Monobehaviour script.

ControllerPhysicsVolume2D

MonoBehaviour class, used in conjunction with controllerParameters2D, which is not.

ControllerState2D

Keeps track of the different states that the controller can be in through properties for raytracing.

FireControllerLevel3

Used to control the fire's vertical velocity in level three. Translating an object in the Y-axes with positive values.

IPlayerRespawnListener

To be invoked by any object that implements this interface, when the player is respawned in the checkpoint the object currently is under.

Player

Main script to handle the Input and tell the CharacterController2D what to do. Also, handles any damage or health that the player receives or any actions such as picking up hay.

PlayerBounds

A simple script to restrain the player and camera's movement. Here we can set the needed behaviour that we want went the player tries to exit the bounds. To either constrain, kill or do nothing.

SimpleEnemyAI

Script to handle basic AI lion behaviour, using raytracing. Specifically, used for the lion to track the movement of the player while being nearby in order to attack and apply damage or take damage and give points to the player. The Circus Director is also using parts of the script.

SimpleGiraffeAI

Script to handle basic AI giraffe behaviour, using raytracing. Used to force the giraffe move towards the player when he is carrying hay.

SimpleGiraffeEat

Script to interact with the player when carrying hay. Triggers animation and a series of events.

Level Effects - containing level mechanics and interactions (damage, checkpoints, etc.)

AutoDestroyParticleSystem

Destroys particle system after one play through.

AutoStartParticle

Start a particle effect.

CenteredTextPositioner

Positions information text in the center of the screen and moves it towards the top using Lerp.

Checkpoint2D

Invoked by the LevelManager when the player hits the checkpoint. Respawns the player to the transform of the checkpoint, based on the X and Y position. Also by using the IPlayerRespawnListener we can handle any objects that we need to respawn when the player does so.

CheckPointVFXScript

Destroys the checkpoints VFX

DeleteObject

Destroys an object after a short period.

EnableRenderer

Used to enable the hay renderer when the player picks it up.

EndingGame

To end the game after the player kills the director and trigger a series of events.

EndOfLevel3

Reaching the last checkpoint of the game *elevates* the player up to the final boss.

Fading

Used to fade in and out every time that a level changes.

FinalText

Displays a thank you note for playing the game and returns to the start scene.

FinishLevel

Transports the player to the next level.

FireballScript

Makes fireballs fall from the sky. (Used together with Firecurtain script)

FireCurtainScript

Spawns fireballs and translates them in the Y-axes. Also adds a random range in the X-axes. (Used together with Fireball script)

FloatingText

Instantiate the Floating Text component on the Floating Text gameObject

FollowObject

Script to show the health bars which object to follow.

FromWorldPointTextPositioner

Used to calculate where in the screen the text should be, based on its world coordinates.

GiveDamageToPlayer

Used to knockback and give damage only to player object.

GiveHealth

Used to give health to the player.

HealthBar

Script to visualize the health percentage of the character in the game.

HealthBarEnemy

Same as above script, except it also destroys enemy's health bar when health is zero.

IFloatingTextPositioner

Interface to be implemented by any scripts that need to have this effect.

InformationText

Displays informational text and plays sound clip.

InstaKill

Instantaneously kill the player.

ITakeDamage

Interface to be implemented by any objects that receive damage.

PathedProjectile

Script that handles the projectiles should instantiate, their destination and speed. Also, handle any damage taken from other projectiles or given to the player.

PathedProjectileSpawner

Script that handles the fire rate of projectiles. Also, drays gismos for usability.

PlaySound

Plays sound when colliding with player.

PointBacon

Adds all necessary behaviour to bacon pickups like adding points, playing an animation and an effect, etc. In addition, implements the IPlayerRespawnListener interface.

PointBallon

Adds all necessary behaviour to balloon pickups like adding points, playing an animation and an effect, etc. In addition, implements the IPlayerRespawnListener interface.

Projectile

An abstract class for all the different types of projectiles. This way we can handle any collision and triggers with all different kind of projectiles.

RotatingCirle

Instantiate a visual effect for the checkpoint.

RotatorCheckPointScript

Rotates the checkpoint parent. (Needed, as the checkpoints are different objects.)

SimpleProjectile

Adds behaviour to the simple projectiles, used by player.

StopElevatorSparkles

Destroys gameObject after a short period.

SubtitleTextPositioner

Places floating text at the bottom of the screen for a limited time.

Managers, handling the game states (start, pause, levels, GUI, etc.)

GameHud

Displaying GUI text for points, bacon and balloons.

GameManager

Adds and resets points though an instance (singleton). With this design pattern, we restrict the instantiation of a class to one object.

LevelManager

Used to control the overall functionality of the game, eg. Where the player respawns, saves points, debug function and manages general messages. Also, handles level change and fading.

Pause

Pause menu functionality.

StartScene

Start menu functionality.

Platforms - (moving, vanishing, trampoline, following path, etc.)

AppearingBarScript

De/Activates platforms whenever the player jumps.

DirectorFollowPath

Defines the circus director's flying swoop path.

FollowPath

Defines the moment pattern for moving platforms.

JumpPlatform

Boots the players Y velocity and plays a sound. Used for trampolines.

MovingPlatform

Creates platform gismos and translates the platform between fixed points.

PathDefinition

Defines a path and creates lines between the points.

PlusChanger

A second iteration of the AppearingBarScript.

TrampolineAnimationScript

Applies animation to the trampoline. Needed, as different object.

vanishingPlatform

Applies a blinking feature to the platform and destroys itself after a while.

Sprite animation

We created our own animation in 24 fps for the character, the Knife Throwers and the Circus Director. We applied different animation for the main character, running, jumping and standing idle.

b) Implementation issues

We used much more time than expected as we were very interested in the developing of the game. Any issues that we came across are mentioned in the technical challenges section and the comment summary.

4. Analysis

a) Playability of the game

The playability of the game is characterized by different *attributes* and *properties* to measure the video game player's experience [8]. Any players/testers trying out the game, including us the developers; revealed high *satisfaction* levels for different sections of the game. The degree of gratification begun rising from the main characters movement and jumping animations and the basic sounds we have used for it.

Also, a handful of people found sections of the game, like the jump related platforms challenging to master but still enjoyable. Of course, these comments are always subjective to the specific tester's capabilities. As a result, the *learning* attribute for the game starts at a basic level for the gamer to understand and use the game mechanics; still it exponentially increases through the different levels and tasks which can be frustrating and annoying.

Moving on to the *efficiency*¸ our game has always been able to catch the player's attention from the first instant, and provoke him to continue playing till giving up actually, as the game's difficulty is increasing through the levels. However, although they can be annoyed by it, the general impression is positive.

The *immersion* level in our game was also taken into consideration as the player needs to be quick and agile through the level with many different tasks and virtual objects. They need to eat bacon and at the same time try to run fast to catch all balloons while trying to understand how the level works. Also enemies that are introduced in level 2 keep the player to their toes and excited without overdoing it. Generally we believe we kept the equilibrium between the proposed challenges and the necessary player abilities to overcome them, in a level high enough for active gamers.

The *motivation* of the each individual player who tried the game varied. Although the set of resources to ensure the players perseverance in the performed actions to overcome the game challenges have been more than enough with infinite lives and throwing knifes, there have been a couple of people that quit from the completion of the game as they believed it was too hard. To our defence, any active gamer that tried the game laughed, got pissed and enjoyed the game. Still this is maybe something we need to take into consideration in the future, based always on our target group.

Emotionally the game stands very close to the motivation attribute, as the involuntary impulses and responses originating from the stimulus of the video game have been through many different states in a short amount of time. Swapping from happiness, excitement and intrigue to fear and sometimes even giving up. This trial and error approach caused much laughter to the testers. We believe this is a success for our prototype.

In our game the *social factor* is something that was unfortunately left out due to time reasons. It is a complete standalone 2D platformer with no cooperative or collaborative way. Of course, we could build on our approach with the points, bacon and balloon system, to have a competitive point system where people compete against each other for the most points. Also developing new shared challenges

that help players integrate and be satisfied with the new game rules and objectives, and motivate themselves to overcome the collective challenges.

Continuing with the *game facets* and the *intrinsic playability,* in our game the rules, goals, objectives and rhythm are being slowly introduced to the player through the levels. As a result, the player can be introduced gradually to the game's nature without being overexposed.

The *mechanical playability* and *interactive playability* have been always in our mind as we find them extremely important. While the player traverses through the levels, information text appears to his aid. In addition, we had a few sounds recorded to play at the same time in order to intensify this, and help the player even more. The information text and the sounds though, have been kept to a minimal so that the game does not become unbearable. Also, we choose not to record any more sounds in the current time frame.

Although the *artistic playability* is not graded for this report, we have taken it into consideration as it gives an extra dimension to the game. For the visual graphics, we created our own sprites and animations. Furthermore, we found sprites and texture online matching to our theme. The sound effects and melodies that we used were also found online, with always keeping in mind our themes and trying to elevate a happy and enjoyable environment. The storytelling part is highly linked together through the levels as the player learns the basics of the game outside of the circus. Then is introduced to extra features inside the circus but unfortunate events and fire spitting lions, force him to run for his life out of the circus.

Finally, we believe we have kept the "global" playability high enough to guarantee the best player experience when the player plays the video game, given our limited timespan.

b) Summary of feedback

Person A

- Good-looking game, I was almost about to destroy my keyboard.
- I think the last obstacle [disappearing platforms in level 1] is way too hard.
- The little man [the player character] is sometimes stuck, if you jump "into a corner". *(fixed)*

Person B

- I also experience the player character getting stuck in corners. *(fixed)*
- It would be nice, if the player character accelerated, because the little man really moves fast! :D
- If you enter the esc menu [pause menu] and press the mouse button, you can see the [player] character throwing a knife in the background. *(fixed)*
- Sometimes the bacon does not reappear, when restarting (I haven't taken a checkpoint or anything) *(fixed)*

Person C

- Well done guys, really enjoyed it! :D
- I got stuck in the giraffe part. *(fixed)*

Person D

- Did you make all this!? It looks awesome!
- I found a bug in the fire! *(fixed)*

Person E

- Really like it, still would prefer to jump with ‘W’ *(fixed)*

Person F

- Good job. Clouds in the front look a bit weird. *(fixed)*
- No restart level in menu. *(fixed)*

Person G

- Awesome work! Maybe the enemy knifes. *(fixed)*
- No restart level in menu. *(fixed)*

These comments were repeated from almost all the people that tested our game. We had approximately 20+ testers and tried to solve all of our bugs iteratively.

5. Conclusion

a) Discussion of the evolution of the game concept

We initially decided to make a 2D platformer with some sort of a shooting element. The immediate thought was a take on Super Mario Bros. combined with a yet unknown shooting element. The context came at a much later date. We first focused on the environment interactions; different types of platforms and bacon to collect. One expressed desire to use the character from the online comic Cyanide and happiness and the idle animation for the main character lay finished shortly after.

The first elements of the game was the ground, the bacon element and pause menu of the GUI, a first attempt on the moving platform and a first attempt on the trampoline. Here we tested character movement thoroughly. We quickly ran into some technical difficulties with the trampoline, but even worse with the moving platform – which turned out to be much more complicated than anticipated.

Running and jumping animations was added, and we decided to remove work focus from animation completely, for the time being. Our own initial experience of the animations so far, led to a goofy, light-hearted and funny theme. At this point the circus theme emerged. We felt it built natural on our current, light-hearted concept, and with trampolines and platforms.

We drafted the first level and added spikes, balloons, checkpoints and an expanded GUI. It's around here that we decide to redo all scripts - to restructure our architecture, which would, among other things, enable us to implement moving platforms properly.

So we created one level for each of us to play around in and test scripts and gameplay mechanics. But it didn't take long before we're all involved in all 3 levels. From here on we more or less worked rather structured until the final version of the game.

b) In hindsight

Architecture

We decided to redefine the entire architecture of our game scripts in the middle of the project because it was not structured well enough. Had we known this, we would of course have implemented the current architecture from the beginning.

Design

We would also have designed the game in much greater detail to begin with, to make sure there would be more cohesion in and between levels. The reason why we have 3 levels is entirely because we thought we could then each work on our own level. However as it turns out, we all worked on all levels.

Planning

It would have been a good idea to use the first weeks solely researching and planning. We more or less just jumped right ahead and created a little here and there – because we were so excited about it! We did start out with some research and planning of course, but maybe we should have been better at delegating assignments with deadlines, right from the beginning – we should have elected someone to be the lead.

c) The next steps

Advanced AI

There's definitely something to be desired in the form of a more advanced AI. Neither of us has had any experience in AI, so it something we would like to look in to, in the future. Currently we have 3 enemies using AI. The most advanced AI is the lion, which spits fire when the player is within range, turns around if the player tries to sneak up from behind and otherwise patrols its territory.

Graphics

We would also like to polish the sprites, to add to the graphical feel of the game. Bringing soul to a game world is an important part of the gameplay. The designers should do their best to make sure there's graphical cohesion, the player must be able to see and feel that all the graphics are “of the same world”. It adds to the gameplay.

Character movement

There's a little bit of player characters movement to polish. We did of course test and polish the character endlessly; however a last fine tuning would be nice.

Level design

We have all tried to make our own games before, but this is our first true production of a game. This however also means that none of us have really given much thought to level design. This is therefore another thing we would look into. After the player character has received a fine tuning, we would redesigning all levels, to make sure they complement the exact player character movement and provides a fluent gameplay and learning curve from level to level.

High score

Adding a table of high scores would be an obvious feature to add. Unfortunately the feature just missed the deadline for our project.

Sound

It could be nice to create even more voices and a find a music score for the big boss battle in the end.

Socializing

Finally, we can make this game more challenging by creating social events and activities that people could compete for and against, such as high scores or items needed to be found, hidden in the levels.

6. References

[1] http://en.wikipedia.org/wiki/Super_Mario_Bros

[2] http://en.wikipedia.org/wiki/Metroid_(video_game)

[3] http://en.wikipedia.org/wiki/Donkey_Kong_Country

[4] http://en.wikipedia.org/wiki/Cyanide_%26_Happiness

[5] http://www.3dbuzz.com/

[6] http://unity3d.com/learn/tutorials/modules/beginner/2d/2d-overview

[7] http://docs.unity3d.com/ScriptReference/

[8] http://en.wikipedia.org/wiki/Gameplay