Milestone 2: Interim Demo
Fall For Me!

Team Cicisoft

Andrea Solanas de Vicente

Ankur Deria

Michael Dey

Bendegúz Timár

Critiques

In this section, we reflect on the critiques from other students.

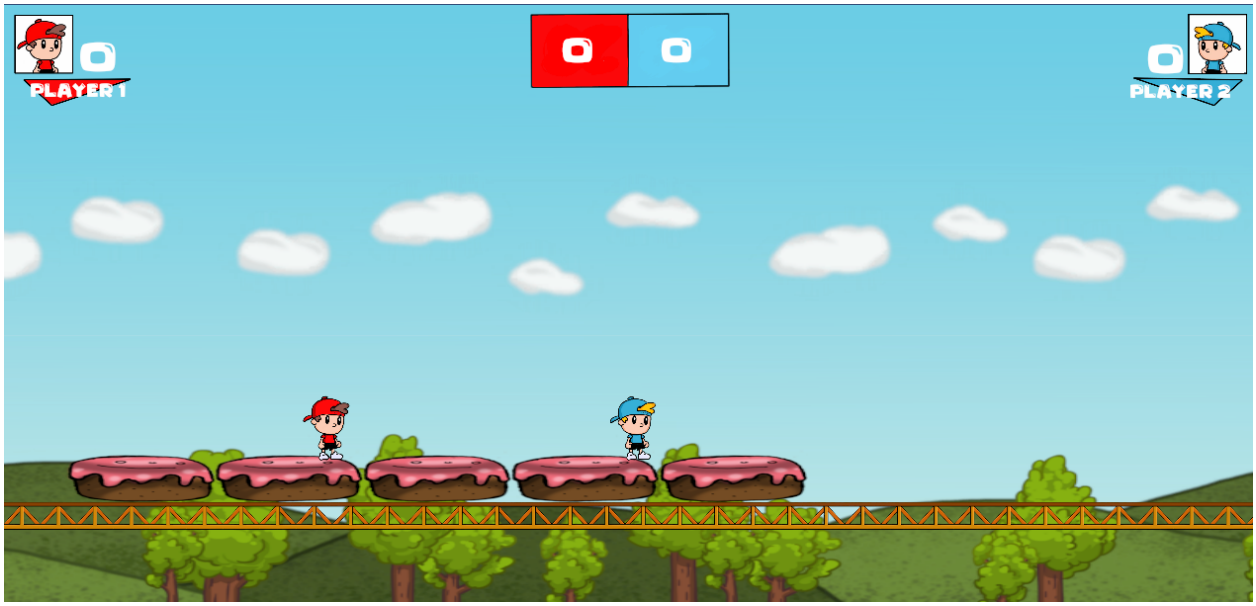
Critique	**Reflection**
Frustration when killed	Timeout is the only reward for pushing off. It would be frustrating to have opponents come back too fast. In the future, we can do a hanging mechanic where the other teammate has to help the player come up to revive. → This is not set in stone → Possibility: Respawn at any base
Better fighting mechanics	The game's focus is on fighting and balancing on the roller coaster. → Focus may shift towards fighting later
Team Combo in fighting	This is a high target. Simple combos are possible because of the current implementation.
NPCs	This is (now) part of the extras

Task progress

Our goal for the interim demo was to produce a first working version of our game that implements most of the main gameplay of the final game. In order to reach this goal, we first wanted to implement the mechanics of the rollercoaster, the gameplay and controls of the characters, add the collectibles and a basic UI.

Basic UI design

For the UI design, we decided to base our ideas on the video games we mentioned in the previous report. There should be a main menu where the players can choose the mode of the game (1v1 or 2v2). For this report, we were not able to add a way for the players to choose their teams just yet, but that will be the next improvement. After the main menu, the game will start with the number of players that have been chosen. Each player has a specific counter where their candy will be shown. Then there is an overall counter for both teams. This counter will be updated when the candy is dropped off at the bases.



For the main menu, we created a simple logo with the name of the game. The player can then choose between the two options of playing, as explained earlier.

Background

The backgrounds are static but their UVs are offset with respect to time and the game speed to make it seem like they are moving from right to left. The backgrounds are separated into different objects, for example, the mountains, clouds and trees. This is done so that they can move at different speeds to give the viewer a parallax effect. The current background is not the finalized one, as we plan to make it look like a theme park by adding a few amusement park attractions. We may even add simple animations to these attractions. Given enough time we

might make these attractions interactable, for example having a ferris wheel close to the roller coaster serving as additional platforms for a short amount of time.

Handling player input

The player input is processed using Unity's new input system. The basic idea behind the system flow is incorporating so-called Action Assets, a mapping of input events to event handler groups called actions. These actions are totally arbitrary, and only have semantic meaning to the developer. Two Action Maps are created, one for the gameplay and one for the menu; this enables registering different handlers in different scenarios for the same input events. The Action Assets are wired into Unity's Player Input component, which automatically creates and exposes an interface of the related action handlers to be implemented in code.

Since both movement, punching and jumping are physics-based, the event handler methods only set action flags for the respective player. These flags are then checked in the FixedUpdate method to ensure that movement physics is accurate and calculated in a fixed timerate. There, punching and jumping is implemented using impulse forces, while movement is a direct smoothed modification of velocity.

Character animation

The animation of movement is done using sprite sheets and Unity's Animator component. We drew a number of frames for all the types of movement, and the animations themselves are rotating these frames around. We created a central Animator component that is the same for each player, defining the animation states and the transitions between them. To be able to use different sprite sheets for the different characters, each character type has its own Animator Override Controller. This component is coupled onto the central Animator, and redefines the animations used in the respective states to achieve different looks for each character.

Collectibles

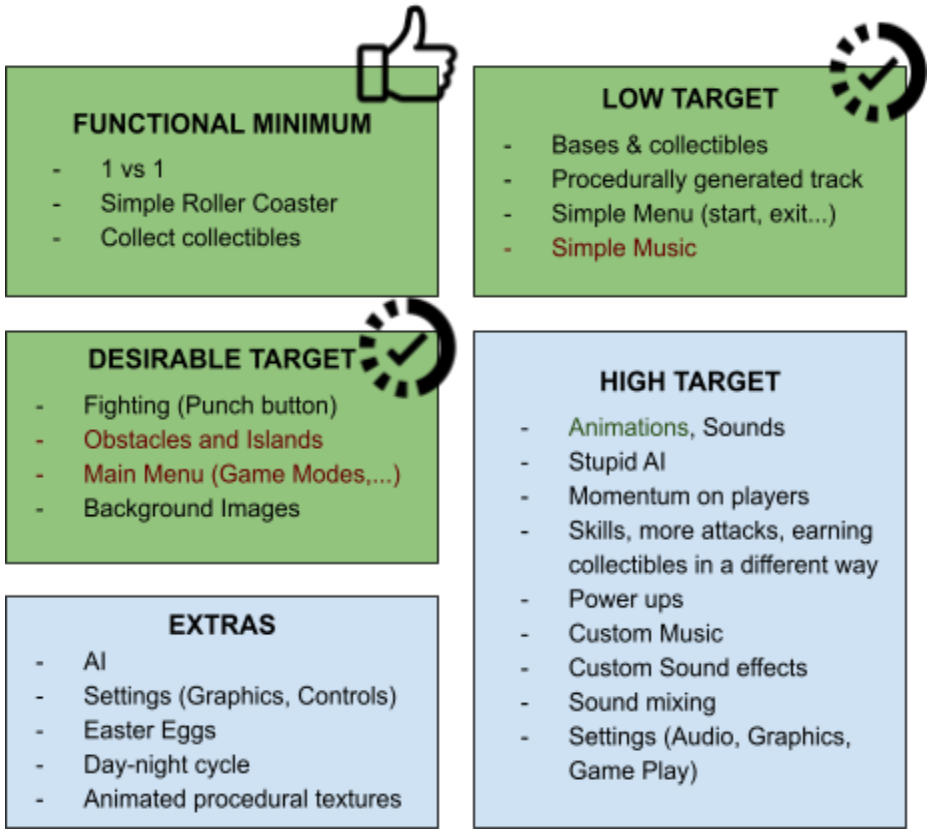
The collectibles are spawned along the track every time a new track segment is added with a 70% chance of spawning. The speed of the collectibles is tied to the track, i.e., if the track moves fast, so will the collectibles. On top of this, the collectibles can have a movement curve independent of the track, for example falling from the sky or levitating along a sinus curve. Spawning in a new object every time would be a huge waste of resources. Instead, an object pool of collectibles is used, where the newly activated object gets its translation set. This is an elegant solution, since we control the spawning rate of the collectibles, and therefore can give an estimated upper limit for the size of the object pool. The players interact with the collectibles based on simple 2D colliders. The colliders are set to be only triggers, which shoot events related to entering or leaving the collider's area, but have no physical consequences, such as pushing the player away when they touch the candy.

Procedural generation of the roller coaster track

For the procedural generation of the track, we used cubic bezier splines. When the left-most point of the spline leaves the screen, a new point is added to a random location on the right of the spline. The distance between the points remains constant as the random point is sampled from a unit sphere's arc (angle constraint). The curvature of the track can be adjusted using the tangents of the spline. The entire track moves from right to left and the Y position is adjusted so that the track always stays at the center of the screen irrespective of if it's going up or down.

Layers

With all these accomplishments, we were able to achieve our functional minimum as well as parts of the low target and desirable target. We decided that some elements of the desirable target were more important than others in the low target, like the music and some artistic elements. The mechanics of the game were prevalent over these aspects.



Problems

Jittering Cars

The first method we tried was to animate the roller coaster using the spline offset value from the nearest point on the spline. But as the spline was being changed repeatedly, the offset value was fluctuating as well, resulting in the jittering of the roller coaster.

The next method we tried was adding an edge collider along the spline. This removed the jittering and the roller coaster could travel smoothly along the spline now. But once in a while, the cars would fall off due to the limitations of Unity's physics engine.

Finally, we solved it by taking into account the changed spline and also the previously calculated nearest point on Spline was in local space. So now that we have the nearest point on the spline, we can align the roller coaster car to it using its tangent and normal vectors. Animating the roller coaster this way not only made it travel smoothly along the spline but also ensured it would never fall off the track. We also add a force along the direction of the movement to make it look more dynamic.

Gravity on the roller coaster

Once the “nearest point on the spline” method worked, the roller coaster moved with constant speed. This was not realistic.

To address this issue, we initially attempted to reintroduce Rigidbodies and gravity into the simulation for a more realistic effect. However, we soon realized that this approach was overly complicated and didn't yield the desired results.

Our final solution is quite elegant. We sum up the angles of the forward vectors of all cars against the horizon. If the sum is positive, the cars on average travel upwards. Based on this value, we made adjustments solely to the background speed and not the rigidbodies. This approach resulted in a more realistic physics simulation, especially when considering that all cars in the roller coaster have the same mass, which is an assumption we made for simplicity.

Future work

Implementing different controlling schemes is definitely on the roadmap. In the main gameplay scene keyboard and controller input will need to be handled at the same time, since we have to imagine that some of the players will be playing with the keyboard while others with separate controllers. Additionally, the keyboard may be set up to support two players simultaneously. This is allowed by the very simple controls, and advised in order to achieve a more accessible user experience. The menu will need to be controlled by either controllers or keyboard. Currently the menu only supports mouse input. Instead of this, the menu will need to be able to change between being controlled by keyboard or controller. Furthermore, hint texts of the controls of the menu may be added onto the menu scene, which texts should follow the currently active input scheme.

In the future, we want to introduce a drag force on the players as they jump on the roller coaster, based on the speed of the roller coaster. This would limit the players jumping around too much and make them only jump when needed.

At the moment, our base spawning mechanic is not very stable. Players do not have much time to stay on the base. We plan to address this issue by changing the base spawning mechanic so the base moves along with the roller coaster for a fixed time, after which it goes away. This would give the players enough time to get on the base, register their collectibles and get back on the roller coaster without slowing down the roller coaster.

We will need to thoroughly playtest character fighting to fine tune the parameters and see what works best, and possibly come up with some combo ideas. We may add a slight upward force to the punches, which would allow the players to do a combo on an enemy. The upward force would stack if the enemy gets punched repeatedly.

A 2 versus 2 players mode is planned to be added. Adding such a mode is a bit more complicated than just spawning in two more players. We will need to fine-tune the spawning rate

of the collectibles to balance out the fact that there are more players in one team, and maybe the length of the roller coaster too.

Work log

Week	Goal	Tasks	Assigned to	Hours expected	Actual hours
Week 4: 01.05 - 07.05	Artwork	Basic character & collectables	Andrea	10-15	12
		Basic rollercoaster	Andrea	5	2
		Simple menu artwork	Andrea	3	3
	Roller coaster mechanics	Moving roller coaster	Ankur, Mischa	10-12	11
	Simple Menu	Start of the game	Mischa	2	2
Week 5: 08.05 - 14.05	Character mechanics	Basic controls	Andrea	10-15	12
		Interaction of characters	Bende	15	14.5
		Characters on roller coaster	Bende	12	10
Week 6: 15.05 - 21.05	Collectables	Availability to collect	Ankur	2	2
	HUD	Food Counter	Mischa, Andrea	6	6
	Simple music	Gameplay track	Bende	6-8	7
	Prototype	Finish and test	All	6	8
Week 7: 22.05 - 28.05	Adding new game mechanics	Bases (make artwork)	Andrea	5	6
		Procedural track	Ankur	10-12	14
		Fighting mechanics	Ankur, Bende	4-5	4
	Simple music	Menu track, Game over track	Bende	6-8	10
	Milestone 2 documentation	Document	All	4	5
		Presentation	All	5	5