Battle of Origins — Interim Release

Patrick Misteli, Ruben Kälin, Jacqueline Staub, Gregory Wyss

April 20, 2015

Contents

1. Current Stage

We have implemented the functional minimum and are nearly done with the low target of our project. The following sections will describe our progress and our difficulties.

1.1 Task Distribution

See Table 1

Task	Description	Who	Hrs	Actual
	Idea Finding			
1.	Brainstorming Design	All	5	7
2.	Character modeling	Greg, Jacq	20	25
	Assignments			
3.	Project Proposal Draft	All	10	10
4.	Prototype Chapter	All	10	10
5.	Interim Report Chapter	All	10	10
6.	Alpha Release Chapter	All	10	
7.	Playtest Chapter	All	10	
8.	Conclusion Chapter	All	10	
9.	Demo Video	Patrick	50	
	Presentation and Demos			
10.	Pitch of the Game	All	7	7
11.	Formal Game Proposal	All	10	12
12.	Paper Prototype	Jacqueline	5	6
13.	First Playable Demo	All	30	50
14.	Interim Demo	All	50	80
15.	Alpha Release Demo	All	100	
16.	Play-test presentation	All	75	
17.	Final Public Presentation	All	40	

Table 1: *Task allocation* Green: Completed

1.2 Project Management

See Table 2

Task	Description	Who	Hrs	Actual
	Functional Minimum			
18.	Players from two teams running around	All	15	15
19.	Level Design: Overflow flat Map	All	15	7
20.	Counting collective hits	All	15	8
21.	Game finishes after 8 min	All	15	10
22.	Winner is Team with most hits	All	15	14
23.	AI Controlled Allies/Enemies.	Ruben	15	25
	Low Target			
24.	Audio: Music + Sound Effects	Patrick	15	2
25.	Physics: Players flying away when hit	All	15	10
26.	Physics: Cooldown before being able to move & attack	All	15	17
27.	Physics: Immunity cooldown before being vulnerable again	All	15	13
28.	Wonder: Wonder is generated after every 50 collective hits	All	15	24
29.	Wonder: Wonder is (visually) possessed by a human player	All	15	10
30.	Wonder: Wonder can visually be cast	All	15	12
31.	Wonder: Wonder converts players	All	15	16
32.	Wonder: Converted Human player plays for the other team	All	15	5
33.	Winner is the team with the most members	All	15	20
34.	Level Design: Map includes obstacles	All	15	7
	Desired Target			
35.	Characters visually polished to look from same theme	Jacqueline, Gregory	15	
36.	Wonder Creation: Creating a wonder by standing together and pressing "commit"	All	15	
37.	Wonder Creation: Cooldown after releasing "commit"	All	15	
38.	Wonder Creation: Increased vulnerability during praying and cooldown	All	15	
39.	Wonder Creation: Larger praying/studying circles will generate quicker progress	All	15	
40.	Wonder Creation: AI upgrade to take wonder creation into account	All	15	
	High Target			
41.	Converted Human player will control free NPC if available	All	15	
42.	Players evolve numerically according to their actions (Running, Shooting, Praying/Studying)	All	15	
43.	Players evolve visually	All	15	

Table 2: *Task allocation* Green: Completed, Yellow: in Progress

1.3 Timeline

See Table 3 and Table 4

Task	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
	Idea Finding													
1.	A	A												
2.	G	G												
	Assignments													
3.		A	A											
4.				J	A									
5.						A	A	A	A					
6.										A	A			
7.												A		
8.													A	A
9.													A	A
	Presentation and Demos													
10.	A													
11.				A										
12.						A								
13.									A					
14.											A			
15.												A		
16.														A
17.														A

Table 3: *Timeline*
A = All, P = Patrick, R = Ruben, J = Jacqueline, G = Gregory

Task	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
	Functional Minimum													
1.					A									
2.				A										
3.						A								
4.						A								
5.						A								
6.				A										
	Low Target													
7.						P	P	P						
8.						A								
9.						A								
10.						A								
11.						A								
12.						A								
13.						A								
14.						A								
15.							A							
16.						A								
17.						A								
	Desired Target													
18.								A						
19.							A							
20.								A						
21.								A						
22.								A						
23.								A						
	High Target													
24.									R	R				
25.									A					
26.									A	A				

Table 4: *Timeline*
A = All, P = Patrick, R = Ruben, J = Jacqueline, G = Gregory

2. Obstacles and Revisions

In this section we will explain some of the difficulties we encountered in the areas we are currently working on. A few of them led to design revisions which are also explained in this chapter.

2.1 Graphical Aspects

We bought two characters including some animations from the asset store. The plan was to adapt the two models to make them more cartoony which turned out to be more difficult than expected.

2.1.1 Blender Integration

Integrating the characters into Blender turned out to be more difficult than expected. Both characters were already rigged when we bought them. After the import procedure the whole skeleton was scrambled and the bones where oriented the wrong way (see Figure 4).

Solution: Redo the rigging and the animations.

2.1.2 Importing into Unity

Importing the animations into unity lead to unexpected issues such as wrong scaling during animations.

Solution: Manual scaling

2.1.3 Looks and Movement

In addition to making them look less lifelike they should also look and move as characters of the same theme. This is only possible with good collaboration of the team members responsible for the modelling. The current state is shown in Figure 1.
Solution: Increased team communication

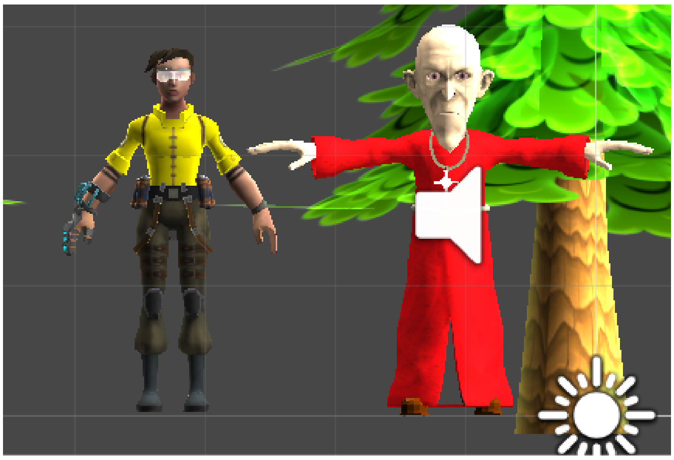

Figure 1: Visual appearance of monk and darwinist.

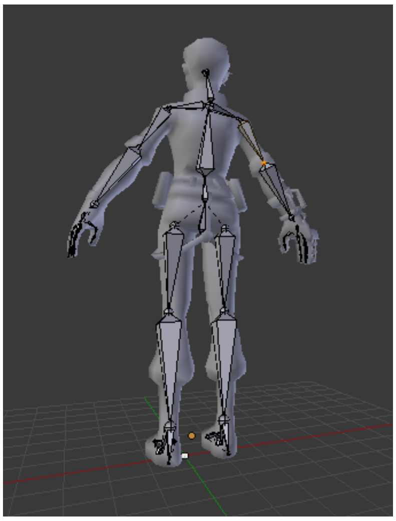

Figure 2: Engineer back

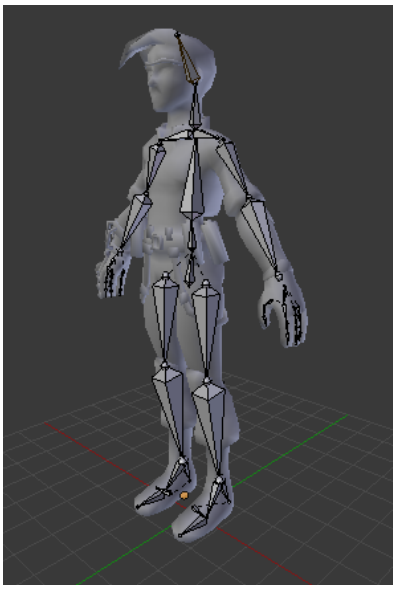

Figure 3: Engineer right side

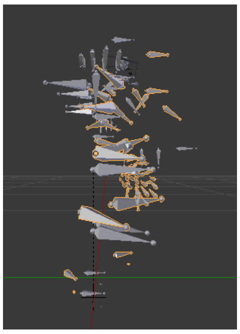

Figure 4: Bones misplaced

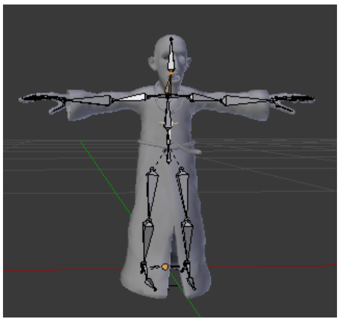

Figure 5: Monk front

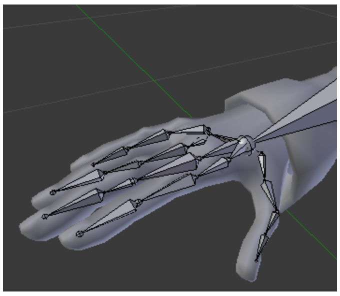

Figure 6: Monk hand

2.2 Artificial Intelligence

2.2.1 Clean Code

We realized quickly that the code responsible for the artificial intelligence (AI from here on) has to be well structured. When working and experimenting with AI this code tends to get messy quickly. This is mainly due to the fact that, the AI needs to maintain a global structure of all Characters, their intentions and actions. Therefore, it does not suffice to handle a Collision in a local manner, but the action needs to be recorded and the strategy of the computer controlled players must be adapted accordingly. In addition, the decisions of the individual players must be explainable with the local knowledge of the individual player.

Solution: Refactoring of previous code, sticking to coding conventions and maintaining an up to date and consistent structure of the game.

2.2.2 Performance

The algorithms executed by the non-human players have to be efficient. Since there can be many non-human players who are executing these procedures very often, we are restricted in their strategic complexity. It is for example unpractical to update the intention of each player in each step. This is, because potentially all players might influence each other player and thus, the number of pairs of players to consider were quadratic in the number of players. With increasing number of players this would not be feasible to compute in each step.

Solution: The intentions of a player are only recomputed when a change makes sense. For example when a player is hit and hurled away he reorient himself. In addition, the exact movement is managed by each individual player, whereas the AI only decides on the intentions of the players (see Figure 7).

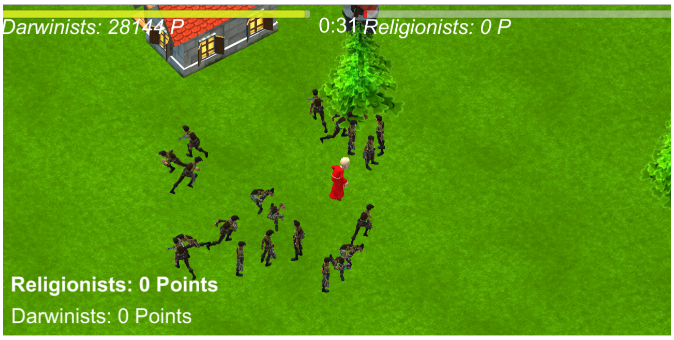

Figure 7: A player surrounded by a lot of non-human players controlled by AI

2.3 Game Logic

2.3.1 People getting thrown off the map

When standing close to the edge of the map and being shot, it was possible to be hurled outside the map. Once this happened it was impossible to return to the map.

Solution: See solution in Subsection 2.3.2

2.3.2 Wraparound Map

We wanted our map to be "Wrap-around". I.e. when you walk out on one side you would walk in on the opposite end of the map. This is necessary to prevent players from hiding in corners, where they cannot be thrown away when shot by an enemy. This would result in a massive advantage when praying or studying because the group cannot be scattered by shooting inside it. Maps with a wrap-around are not natively supported by Unity. Thus, we would have to implement it which would become extremely complicated in terms of many objects which would have to be relocated and reinstantiated at any moment. We would not be able to use solely the physic engine's procedures like for example collision checks, if they would happen in a wrap-around scenario.

Solution: We implemented the map as a island with surrounding water. As soon as a player falls into the water he is respawned somewhere on the map.

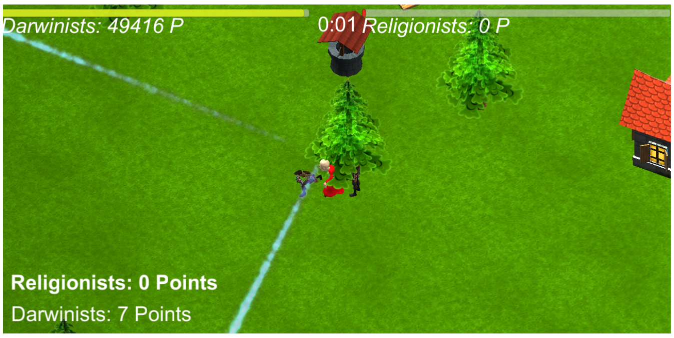

Figure 8: A player being attacked while standing within a tree

2.3.3 Spawning Points

Initially we generated random spawning points. However, sometimes this lead to a scenario where a player would spawn inside some other element, for example a house or a tree. Such a player was not able to get outside of the element and could only shoot from its stationary position (see Figure 8).

Solution: Ask the NavMesh whether the random position is free and on the map.

3. Interim Conclusion

We are well on our way to make this a fun game and while we have had to tackle a few obstacles already we have not run into an issue that would cause us to make severe game design changes.