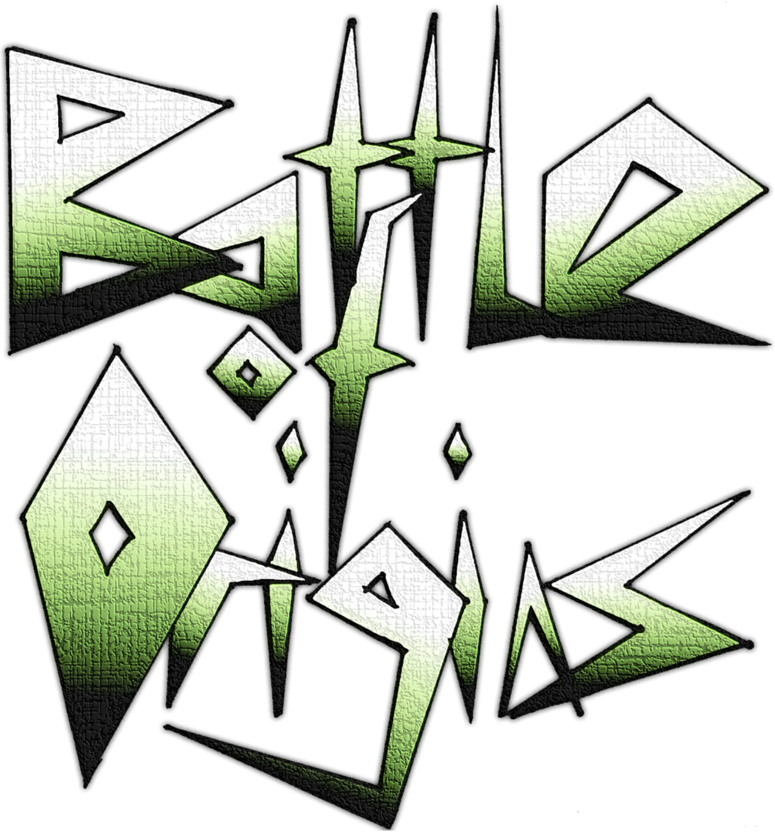

Battle of Origins — Alpha Release

Patrick Misteli, Ruben Kälin, Jacqueline Staub, Gregory Wyss

May 4, 2015

Contents

1. **Introduction** **2**

2. **Current Stage** **2**
2.1 Task Distribution . . . 2
2.2 Project Management . . . 3
2.3 Timeline . . . 4

3. **Obstacles and Revisions** **6**
3.1 Physics . . . 6
3.1.1 Explosion Forces . . . 6
3.2 Sound . . . 7
3.2.1 Animation Sound . . . 7
3.2.2 Walking Sound Overpowering . . . 7
3.2.3 Creating a wonder Sound Overpowering . . . 7
3.3 In-Game . . . 7
3.3.1 Spawning in Buildings/Rocks . . . 7
3.3.2 Camera for Multi-Player . . . 8
3.3.3 Human Creating Wonder . . . 8
3.4 Modeling and Animations . . . 8
3.4.1 Rotated Model . . . 8
3.4.2 Left-Right, Up-Down Issue . . . 9
3.4.3 Unused Animations . . . 9
3.4.4 Scaling after Export from Maya . . . 9
3.4.5 Inverse Kinematics . . . 9
3.4.6 Body parts growing . . . 10
3.4.7 Shadows . . . 10
3.5 AI . . . 10
3.5.1 Praying for Human Players . . . 10

4. **Alpha Release Conclusion** **10**

1. Introduction

This document describes the current working stage (Sec. 2.) and the progress that has been made between the interim release and the alpha release (Sec. 3.). The ideal would be to have a stable alpha release to tweak in game values such as number of players, number of obstacles, time to create a wonder and so on.

2. Current Stage

2.1 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	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	50
16.	Play-test presentation	All	75	
17.	Final Public Presentation	All	40	
	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

Table 1: *Task allocation* Green: Completed

2.2 Project Management

See Table 2

Task	Description	Who	Hrs	Actual
	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	150
36.	Wonder Creation: Creating a wonder by standing together and pressing "commit"	All	15	11
37.	Wonder Creation: Cooldown after releasing "commit"	All	15	14
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	20
	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	30
43.	Players evolve visually	All	15	
	Extras			
44.	Online Multiplayer	All	15	
45.	Procedural level-design (each level is different)	All	15	
46.	Classes of characters (specialized for praying/studying or shooting)	All	15	

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

2.3 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													
18.					A									
19.				A										
20.						A								
21.						A								
22.						A								
23.				A										
	Low Target													
24.						P	P	P						
25.						A								
26.						A								
27.						A								
28.						A								
29.						A								
30.						A								
31.						A								
32.							A							
33.						A								
34.						A								
	Desired Target													
35.								A						
36.							A							
37.								A						
38.								A						
39.								A						
40.								A						
	High Target													
41.									R	R				
42.									A					
43.									A	A				
	Extras													
44.													A	
45.												A		
46.												A		

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

3. Obstacles and Revisions

3.1 Physics

3.1.1 Explosion Forces

When an opponent is hit we want him and the surrounding opponents to fly away. Using the addExplosionForce would sometimes not affect an opponent within reach and the opponent would simply stand still. Furthermore we could not add a y force because of the NavMesh. The NavMesh is needed for the AI to find a path to their new target thus disabling it is not an option.

Approach 1: We tried adding forces manually by calculating the vector going from the explosion origin to all surrounding targets. This did also not give us the desired result since adding a force in one frame would not affect a character enough to be noticed visually

Approach 2: We increased the forces in approach 1 until we saw the characters visually move. The problem now was that the characters moved within one frame thus creating a teleportation effect rather than an explosion effect. Ultimately in the game this does not make a difference since after being hit the character should be moved to a location slightly further away and regain the ability to move after a short timeout. However due to visual aspect we decided to pursue the problem further

Approach 3: After a hint from a teaching assistant we started a new project where we exclusively test explosion effects. Starting from 0 again we could determine a way to have the desired explosion effect without suffering any other major issues.

Solution: Multiple factors have helped us getting the desired explosion effect.

- Use addExplosionForce.
- Make sure the distance of the explosion origin and the character is large enough. If the explosion takes place inside a collider of a character he is only partially affected since some forces cancel each other out.
- Temporarily turn off NavMesh as soon as a character gets hit. This allows forces in the positive y direction.
- Same Mass and Drag for all characters. This brings consistency over all characters.
- Turn off root transformation in the animator.
- Temporarily turn off rotation freeze while in flight. This allows a stronger visual effect then the character is hurled and spun away.

When applying all these items a new problem arises. After a character has fallen a script will attempt to re-enable the NavMesh. However it can happen that a character flies unto a house or rock where the NavMesh cannot be enabled anymore. This was solved in two ways. The buildings have improved colliders (described in Section 3.3.1) which make landing on a building less likely. If it does happen a character is respawn after 10 seconds. Furthermore if the character is near the ground but cannot connect to the NavMesh the character is teleported to the nearest point on the NavMesh. The teleporting distance is mostly small enough to not be visually detectable.

3.2 Sound

3.2.1 Animation Sound

Adding a sound to a animation can be done in two ways. Adding it in the animation directly. This would require adapting all animations and gives very little control over the sounds within the game scripts. A second solution is to add a script to an animation which carries the audio source. For an inexplicable reason this is also not possible as the sound file cannot be loaded within an animation.

Solution: We created a central audio manager that takes care of all the sounds and can be called from all scripts.

3.2.2 Walking Sound Overpowering

Having a walking sound for each character adds to the realism. However hearing the footsteps of potentially 100 characters results in a noisy clutter of sounds.

Solution: The central audio manager keeps track of all walking characters and will play at most 5 walking sounds at the same time.

3.2.3 Creating a wonder Sound Overpowering

Analogically to the problem described in Section 3.2.2 the same problem appears when generating a wonder. Every character can potentially create a wonder resulting in potentially hundreds of wonder-creating-sounds.

Solution: (Same solution as in Section 3.2.2. The central audio manager keeps track of all walking characters and will play at most 5 walking sounds at the same time.

3.3 In-Game

3.3.1 Spawning in Buildings/Rocks

Characters sometimes spawn in buildings. We select a random point on the map but do not check whether this point is inside a building, rock or church. If a character is spawned inside such an object he cannot escape it

Solution: The actual problem was that the NavMesh inside a building was determined as walkable since the buildings had no floor. After inserting a floor and adding a collider surrounding the building characters stopped spawning inside buildings.

3.3.2 Camera for Multi-Player

Camera perspective: In a multiplayer setting, the camera needs to zoom in and out and move around according to the players positions. However, vertically the camera always showed more than the size we set it to.

Solution: The reason for the difference between the vertical and the horizontal axis originated in the fact that our orthographic camera perspective was tilted by 45 degree. Thus, we had to divide the vertical axis by a factor of sqrt(2) in order to get consistency among the axis.

3.3.3 Human Creating Wonder

The implementation of the praying mechanics for human players was difficult, because it is hard to guess the exact intention of the player. If the player wants to pray with an other player who is currently attacking it should fail, whereas if the other player also prays it must work. Moreover, finding all players close by is a costly operation and cannot be executed for each player at each update step.

Solution: NPCs select a target to pray with and efficiently check if this target is eligible for praying and close enough. Since this trick does not work for human players we introduced a commit command. The close players are then only computed for players pressing the commit button. This provides an efficient enough solution that allows also human players to actively pray.

3.4 Modeling and Animations

3.4.1 Rotated Model

Unity and Blender do not use the same axes to to represent the three directions. Exporting a correctly oriented model from Blender to Unity resulted in a rotated model in Unity. This problem seems to be one of the things users of Blender just need to know, wile modelling. We did not find a script to resolve this issue.

Solution: We just rotated the model in Blender before exporting it to Unity. Furthermore we had to be careful to assign the correct up, and forward direction in the exportation process.

3.4.2 Left-Right, Up-Down Issue

After the orientation of our models were correct we encountered the issue, that the model we wanted to use had the opposite top-down, left-right orientation. Since we have two models which should be controlled the same way their orientation should match, which was not the case.

Solution: We resolved the issue by multiplying the X and Z axis by a factor of -1.

3.4.3 Unused Animations

Animations are linked to a fake user in Blender, so they are really hard to delete afterwards. We tested some animations, we ultimately don't need anymore, but they were hard to get rid of since they were linked to the fake user.

Solution: We decided to keep all animations. After the settings are made in the animation controller everything is fine anyway, and they might come in handy someday...

3.4.4 Scaling after Export from Maya

We had the problem that each time we tried to import an animation from Autodesk Maya, while playing the animation the mesh of the character was scaled by a factor of 2.54. After several mails with our tutors and hours of internet research we found that after locking the character definition in Maya for some reason each bone was scaled by the mentioned factor. This problem could not be solved in Maya, because we did not find a solution.

Solution: perform a little hack. As before we import the animations into Unity. Then we copy the animation clip in the FBX file and assign this animation to the animation controller. In the animation window of unity we could find scale properties in all imported animations that scale the whole mesh by 2.54 in all directions. So, we only removed the scaling properties in all animations.

3.4.5 Inverse Kinematics

We used bought models that already included some basic animations. Nevertheless, we also wanted to edit these already existing animations or create new ones and use the predefined as a template. If such an FBX animation is loaded into Maya it appears as a forward kinematics animation with all the keyframes that rotate the joints. We would have liked to define a control rig for the existing skeleton, to edit the existing animations by using inverse kinematics, which is much simpler in our eyes. But when we created a control rig for the existing character definition all keyframes and therefore all animations were gone.

Solution: no use of inverse kinematics (only forward kinematics) when editing predefined animations.

3.4.6 Body parts growing

As the skills of our characters are improving we want to visualize this process by dynamically enlarge hands representing higher shooting power, enlarge feet representing higher running speed and enlarge the head to represent higher praying/studying skills. Our character are built up hierarchically, so our initial idea was to use the localScale property of corresponding transforms. Unfortunately, this had no effect at all and we tried to find some answers in the internet. So, we removed any scaling properties from the animations so that no hierarchical scaling overwrites our changed localScale and also removed the animations to be sure they don't affect the scaling behavior. Then we tried to scale the body parts by first detaching the transforms from their parents, scale them using again the localScale properties and then re-attach them to their original parents. Until now none of these actions did help, we even tried to assign the localScale to itself in every Update() method as it was suggested in an answer of a forum.

Solution: we still don't have a solution at the moment...

3.4.7 Shadows

Leafs of trees we are using in our scene are rendered using sprites and do not cast shadows by default. Like this, only the trunk of each tree cast a shadow which resulted in a weird looking environment.

Solution: We use a custom sprite shader that casts shadows of our leafs.

3.5 AI

3.5.1 Praying for Human Players

4. Alpha Release Conclusion

The problems we face are problems mostly deep in the code. While we do have a robust running system on the outside we strive to perfect and polish the details. These details also include choosing ideal parameters for the game such as number of players, duration of a match and number of human players. This will be approached with intensive testing which will be the next step in the project.