ETH Zurich, Game Programming Laboratory - SS12

Polarity - Interim Report

Group 4: Jakob Progsch, Werner Randelshofer

1. Progress

The table below shows the development schedule from our project proposal.

We are behind schedule with the functional minimum of the game. We needed more time than expected for modeling the player characters. Also, we decided to implement a level editor early on.

Date	Milestone	Work items	Who	Progress
2012-03-19	Prototype chapter			
2012-04-16	Functional Minimum	Scene description file	Werner / Jakob	60 %
		Game loop	Werner	100 %
		Player character 3D object	Werner	80 %
		Scene elements 3D objects	Werner / Jakob	40 %
		Game items 3D objects	Werner	10 %
		Static intro and outro screens	Werner	100 %
		A single playable level	Werner	20 %
		Physics Engine	*Jakob / Werner*	*50 %*
2012-05-07	Low Target	Multiple walls in different 3D orientations	Jakob	80 %
		Playable game items, for example: nail, magnet, ...		
		Multiple playable levels		
		Level screen showing times and collected stars	Werner	100 %
2012-05-14	Desirable Target	A reasonable number of levels including the final level with Ms. Boson		
2012-05-28	High Target	Eye candy: Camera movement, visual effects		
?	Extra Target	Level Editor	Jakob	80 %
		Multi-player game		

2. Gameplay

So far, our gameplay consist of an interactive player character supporting 2D movements along a triangle mesh or a curved bezier surface. See Figure 1 and Figure 2.

The polarity of the player can be changed by pressing the X-key. The color of the player changes from blue to red. The A- and be B-key can be used for cheating. This is a debugging feature, which we will remove for the final version of the game.

The player is rendered as a 3D object floating on top of the game surface.

The player can collect stars (currently visualized as yellow circles) and bump into walls, and finish a level by moving to the exit. In the final game, the stars and the exit will be visualized as 3D objects. The walls will be replaced by a track which is delimited by magnetic bezier tubes and spiky elements.

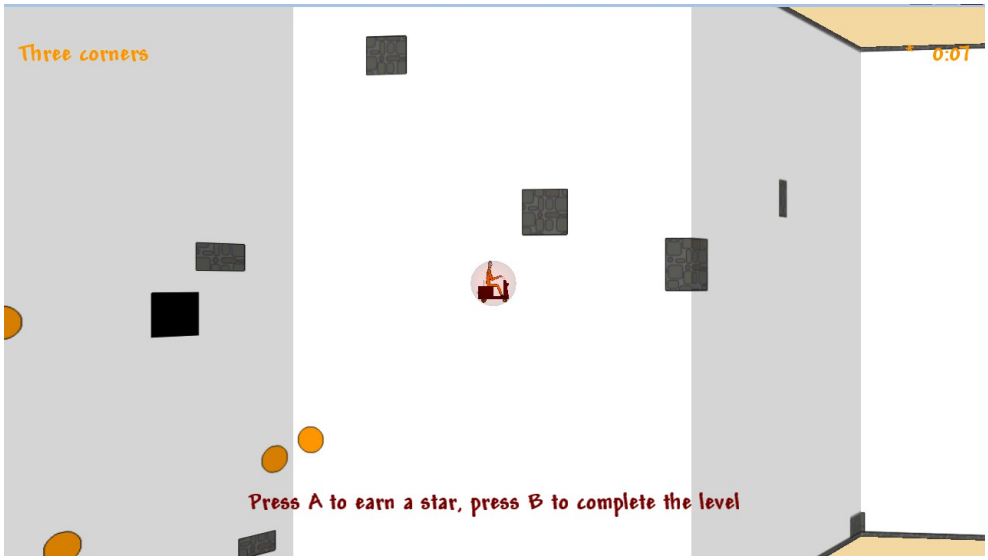

Figure 1: Polarity level on a triangle mesh.

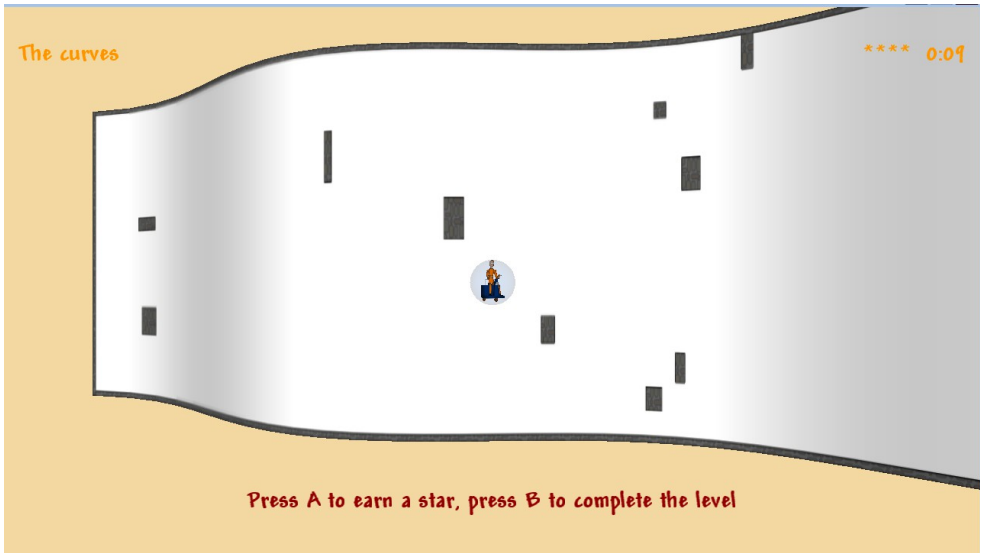

Figure 2: Polarity level on bezier patch

3. Physics

For now, we use the Farseer physics library. This library handles the movement and collisions of the

player and scene elements in a 2D surface. We have an incomplete implementation of our own physics library which will handle movement and collisions in a constrained 3D world. The world is constrained by the surface patches on which the game takes place.

4. Game Characters

We tried out a number of different modeling tools: Blender, Maya and Cinema4D.

After loosing a lot of time with Blender and Maya, we use now Cinema4D for modeling the game characters.

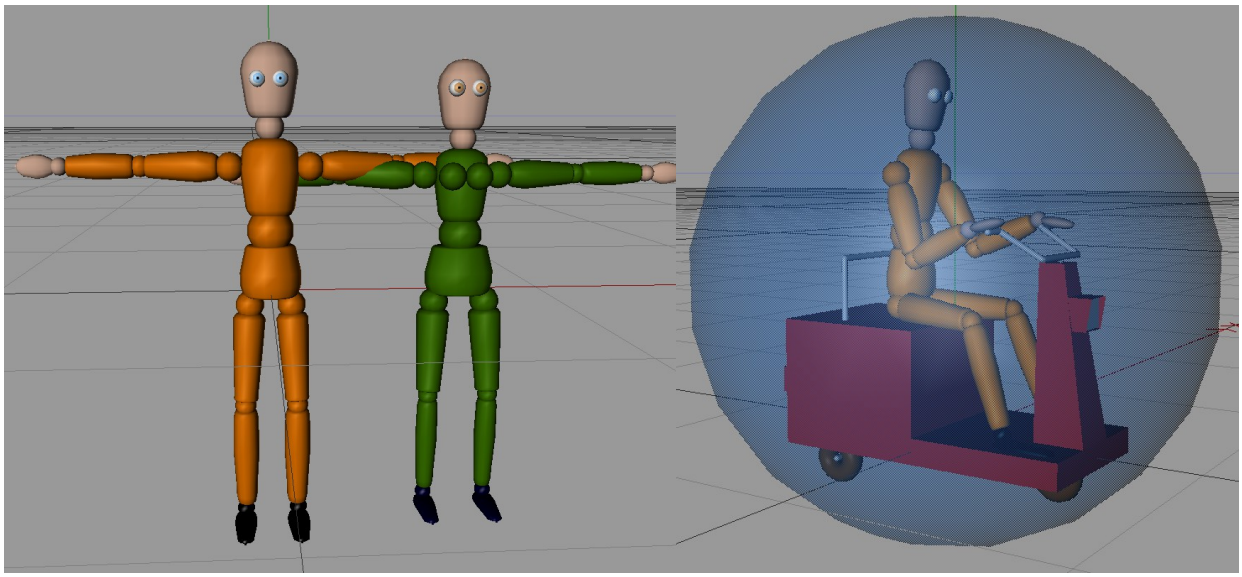

Figure 3: Stick figures created with Cinema4D

3. Level Editor

The level editor is implemented in a separate XNA application. This application can only be used on a PC, because it requires keyboard input for selecting scene elements.

Figure 4 shows a screenshot of a two bezier surface patches and a number of bezier curves on the surface.

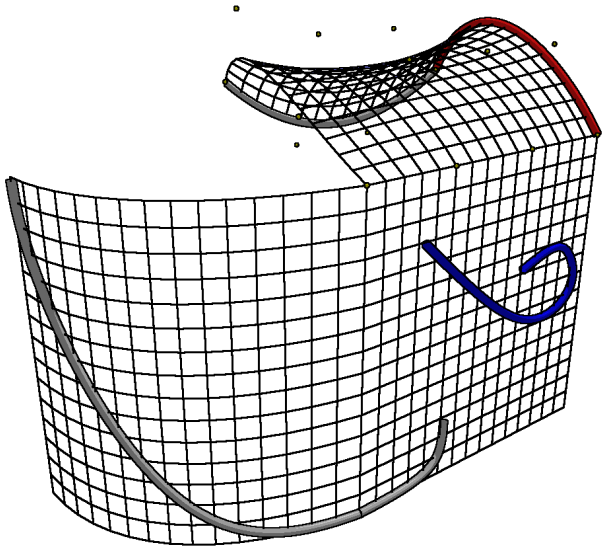

Figure 4: Level Editor