CS 428 Spring 2021

Monster Creator

Elizaveta Belaya · Abhinav Madahar

Topic

Our program, *Monster Creator*, lets the user create a monster. The monster will have one of two body types: a simple cylinder or a more complex horse-like torso. The user adds different body parts onto the monster, like arms, noses, and ears. The user then selects skin texture and other body attributes.

This is similar to character creation systems in popular video games like Spore, The Sims, and Cyberpunk 2077.

Goals

1. **Encourage creativity.** The users will want to create very complex monsters with bodies unlike any of those found on Earth. We want to foster this creativity because it helps relieve stress and soothes the mind. Since many video games create more stress for the player, we want this game to be an oasis.

2. **Make the program simple enough for a child to use.** Children often like to play video games, but not all games are appropriate for children. This game does not have violence, drug use, and other elements which make a game unsuitable for children. We will make this game simple enough for a child to play to encourage children to play this game instead of other, more adult-oriented games. We note that children who played the video game Spore spent more time studying on average [1].

Challenges

1. **Body part modelling.** We wanted to model as many body parts as we could, but we only had enough time for about 10 body parts. We have eyes, noses, and ears. We made these body parts using Blender, which let us write intricate, detailed body parts. We had some difficulty at first importing the objects into Three.js, but we figured it out.

2. **Continuous body shapes.** Animals in the real world have continuous bodies, and there are no gaps in their bodies. We had to make sure that our monsters do not have gaps where different polygons fail to intersect. We ensured this by adding offsets to the body parts programmatically. When the user clicks on the monster to add a body part, the body part is placed exactly where they clicked.

3. **Adding monster attributes.** If someone wants to make a dog, then they would start with a long tube and add four legs, a tail, eyes, and a mouth. However, they cannot place the legs anywhere; the legs must be under the animal. Similarly, the tails must be in the back of the animal and the eyes and mouth must be in the front. Our user must be able to control where the body parts go. We accomplished this by placing the body parts where the user clicked. By clicking in the right places, the user can control where the body parts go.

4. **Animations.** We wanted the users' monsters to move slightly to make the game more lively and fun, but we did not have enough time to do so. This is the only part of the project which we did not reach.

Timeline of Challenge Resolution

Week 1. We used the sample code from homework 2 to start our project. At this stage, our project was 2-dimensional and we used bare WebGL with no external libraries.

Weeks 2 and 3. We had exams, so we didn't work on the project.

Week 4. We drew a stick figure and added a simple arm and leg. The user could click somewhere on the stick figure to add an arm or leg.

Week 5. We added an undo button so the user can undo adding a body part. We then wrote the report.

Week 6. We switched from bare WebGL to Three.js. Three.js is 3-dimensional, so we now had a 3D environment. We started with just a cylinder as a torso, and the user could not add body parts. The user could also move the camera by pressing the C key while moving their mouse around the canvas. We decided to require them to press a key because, normally, clicking on the canvas adds a body part, so we needed a way to signal that the user wants to move the camera. The user can zoom in and out with the scroll wheel.

Week 7. We added body parts, so the user can add body parts by clicking on the torso. They can also add body parts on top of other body parts by clicking on them. The only body part we have at this point is a plain cylinder.

Week 8. We replaced the plain cylinder with an eye made using two spheres, one white and one black. We then add simple arms made out of cylinders.

Week 9. We design a better arm in Blender, which lets us add much more detail. We also learned how to import objects we make in Blender into Three.js. We then rewrote the HTML to improve accessibility for the blind and for people with low vision. We also added an undo button, which works the same as the undo button in the WebGL version. We also let the user change the torso color with three sliders for red, blue, and green. We also set restrictions so that the user cannot zoom too far in or out.

Week 10. We added grass to the ground. We then added a preview feature; when the hovers over the monster, it shows a live preview of the body part so they can correctly place it. We then added several more body parts which we made in Blender. We restructured our system so that we had a list of body parts and their attributes. To add a new body part, we just add a URL to its materials file, a URL to its object file, and a few properties about it. This way, adding new body parts is a matter of modifying data, not code. We also cache the body parts so that, when adding a body part twice, we only load it from its URLs once.

Individual responsibilities

1. **Elizaveta Belaya** designed the monster body parts, wrote the code that imports the body parts, and added things like a ground and the sky.
2. **Abhinav Madahar** wrote the UI and the code that adds body parts to the monster.

References

[1] DorothyBelle Poli, Christopher Berenotto, Sara Blankenship, Bryan Piatkowski, Geoffrey A. Bader, Mark Poore; Bringing Evolution to a Technological Generation: A Case Study with the Video Game SPORE. The American Biology Teacher 1 February 2012; 74 (2): 100–103. doi: https://doi.org/10.1525/abt.2012.74.2.7