CS 174C W24: Corgo in Wonderland

Team members:

- Daniel (Yun) Tsai
- Alex Stavedahl
- Jess Xu
- Rachel (Ray) Hsiao

Our project is "Corgo in Wonderland", a short animation of a corgi (named Corgo) exploring a couple of locations in a surreal world. This concept was pitched by Daniel! The algorithms we implemented are spline interpolation, mass-spring damper systems, collision detection, and articulated dynamics.

The instructions for how to run and explanations of each aspect of our project are detailed in the rest of this document.

How to run

It is **HIGHLY recommended to run the project using the following link**: https://rh5140.github.io/174c-corgo/. To switch between scenes, click the buttons on the top left corner of the webpage.

Only if it is **absolutely necessary**, you can run host and open the project locally in your browser by running the **server.p**y script (double click the file), then navigating to **localhost:8000** in your web browser (Chrome, Firefox, etc.). When opening the webpage, it is **extremely important that the page be hard refreshe**d. The following are key combinations for hard refreshing on the mostly commonly used browsers:

Chrome + Windows: Hold the **CTRL** key while clicking on the refresh button.

Chrome + Mac: Hold the **SHIFT** key while clicking on the refresh button.

Firefox + Windows: Hold the **CTRL** key while clicking on the refresh button.

Firefox + Mac: Hold the **SHIFT** key while clicking on the refresh button.

Safari: OPTION + **SHIFT** + **E**

Spline interpolation

The implementation for spline interpolation can be found in `lib/spline.js`. It is built off Ray's assignment 1 implementation.

Corgo moves along a spline in each scene by updating position, velocity and acceleration with the spline's `get_position` function.

In the rope bridge scene, Jess used the spline to have Corgo bravely walk across a rickety bridge. The spline's control points were determined by the locations of the bridge planks, so they had to be recalculated periodically as the bridge swayed. Corgo's walk speed could be changed by sampling the spline at wider intervals.

Splines are additionally used for the camera control system and the animation of the flower, which will be discussed in subsequent sections.

Mass-spring damper system

The implementation for the mass-spring damper system can be found in `lib/particle_system.js`. It is built off Ray's assignment 1 implementation.

Ray used it in the mushroom scene (`mushroom_scene.js`) for the frog bouncing on the mushroom, with the frog motion following the motion of a particle system connected with springs.

Jess used it in the rope bridge scene (`rope_bridge.js`) to simulate the rope being swept by wind. Spring and damper constants were chosen to make the bridge sway naturally without external wind forces being applied to the entire system. The planks on the bottom of the bridge were also modeled by springs, with higher spring stiffness constants than the ropes so that their lengths would not noticeably change. This approach connects the separate ropes in the bridge, allowing it to sway as one.

Collision detection

The collision detection implementation is the `hit_ground` function inlines 70-75 of `lib/particle_system.js`. Instead of generalizing the collision code, Ray did the hacky method of shifting the whole ground down 3 units in the y-direction since the ground is hardcoded to be y=0. The top of the mushroom the frog is bouncing on is at y=0, so the frog bounces back up when the collision is detected.

Articulated dynamics

The forward and inverse kinematic systems used in this section were based off of Daniel's assignment 2 implementation, found in `lib/kinematic_body.js`.

Corgo's animations are implemented using a forward kinematics system that takes in rotation data. Alex implemented animations by determining angles for the tail and feet over time, which are given to the FK system to create the walking animation.

The flower's animations are done using inverse kinematics. Alex created three splines, which are used to generate targets for the IK to follow. This animation can be seen in `flower_dance_scene.js`.

luid simulation/Basic Rigidbodies

Daniel implemented fluid simulation using a particle similar to the one we made for spring dampener, but the springs were replaced with the attractive and repulsive forces for each fluid particle. To optimize the performance, the attraction force wasn't applied every frame, but every few frames, with a higher attraction force to compensate. To render the fluid, a marching cubes algorithm was used on the particle position, with a semi clear material applied to it to give it the appearance of water. The particles were also able to interact with any rigidbody via point-box collider collision calculations. The code can be found in `lib/liquid_particle_system.js` and `lib/rigidbody.js`.

Camera

Alex used the spline system to move the camera in some scenes. The camera uses either a static point to look at or dynamically tracks Corgo's position in the scene as he moves around. An example of the moving camera is present in `rope_bridge.js`,

Code organization

Daniel modularized and organized the codebase. Daniel also reworked some existing code to switch scenes without crashing by keeping all scenes loaded in memory, and swapping which scenes are displayed at any given time using CSS styling rather than swapping the scenes in memory, which caused WebGL issues. The code can be found in `main_scene.js`.

3D modeling

Alex and Daniel made a lot of models in Blender! For many of these assets, Alex made the initial first pass of the model and Daniel further refined the model. These models include Corgo, the frog, and the flower. This process allowed us to create geometrically complex shapes for our scenes for a better visual experience.

An unexpected issue that came up was that the tinygraphics library has issues with quads, so we needed to make sure to triangulate the meshes.

Texturing

Daniel made textures for the flower, mushroom, and water scene.

Credits for other textures used

- Mushroom floor texture: https://3dtextures.me/2020/03/23/mushroom-top-001/
- Water texture: https://stock.adobe.com/images/shining-blue-water-ripple-pool-abstract-vector-background/250498041?prev_url=detail
- Rock texture: https://stock.adobe.com/images/rock-background-texture-wall-background-stone-abstract/319343596?prev_url=detail
- Wood texture: https://www.pexels.com/photo/brown-wooden-surface-129733/
- Rope texture: https://stock.adobe.com/images/linen-rope-background-yarn-stripe-texture-strong-fiber-lines/377880724?prev_url=detail

Audio Credits

- Mushroom scene background audio: https://freesound.org/people/soundshmyak/sounds/697832/
- Dog bark audio: https://www.zapsplat.com/music/dog-bark-small-2/
- Wind audio: https://pixabay.com/sound-effects/howling-wind-109590/