There are two views for this project, and while both allow for first-person control of the camera, they focus on different things and have differing controls. The free view is how it starts, and allows for full view and control of the simulation (F3 to spawn creatures), and is also best for exploring the environment. The inspect view focuses on a single creature and has a controllable light.

Things of note to see in the project:

- Spawn/death animations done with a combination of shaders and translation/rotation.
- Random terrain generation, with two passes for calculating normals and then averaging them for Gouraud shading (built from scratch).
- Lamp model with light source (bulb goes off when inspect mode light is used).
- Crawler models themselves, which are patterned/colored according to their DNA via shaders, and have several 3D details.

One thing to note, which I go into detail about below, are some memory issues I was encountering. They were rare, and I have code set up to deal with it if it happens, so it is possible you could see something in the logs about that.

Controls

Esc key: quit
w/a/s/d: camera movement
arrow keys: camera look movement

f key: toggle view mode (free view/crawler inspect view)

F1 key: toggle creature movement
F2 key: toggle light auto-movement

In free view:
F3 key: spawn in a new egg in a random position
j/J key: increase/decrease spawn size of crawlers
k key: kill all living crawlers

In inspect view:
+ or - : increase or decrease light distance
] or [: increase or decrease light height
> or < : move light sideways in either direction

j key: activate spawn animation
k key: activate burn animation

Future Work / Un-Remedied Issues

There were certainly a lot of challenges with this project, and unfortunately many were in part due to the short time-frame. I'm happy with what I got done in that time and what I've learned in this course overall, but there were a few issues that I couldn't get fixed in time.

There were strange memory access violation exceptions that would come up, and I spent a great deal of time trying to find the cause and I could not. They were not very common too; I might have ran the same code 20 times before it showed up. It definitely had to do with the VBOs, but I couldn't find anything more specific than that.

The issues with frame-rate persisted, even after I fixed that first issue. And ultimately, I think even the first issue just pointed at the number of polygons being the problem. If I had more time I would have looked into this a bit further and maybe refactored the code to use a more efficient data store.

I also wanted to make the collision detection more robust, but you had mentioned for the proposal to focus mainly on the graphics portions and not the physics, so this problem took a backseat. It works fine as it is, but crawlers can definitely get caught up in each other sometimes, even with a collision cool-down.

I also intended on making the tube/pipe generic object the "right" way rather than using spheres as joints, but it is a fairly difficult task to do from scratch, and I didn't realize this until I was deep into it. I had attempted to try rotating the 3D coordinates of ellipsoids, but I think projections of points onto a plane was the better (and perhaps necessary) approach.

More In-Depth Description of Crawler "Life"

The crawlers begin as eggs, spawned randomly about the area within the boundaries. However, if two crawlers would overlap, the random location is resampled until an open space is found. If there's not enough open space after 100 resamples, they will simply stop spawning. Then after a pre-set time until hatching, the crawlers will "hatch" and spawn in as adults. The spawn effect is done via fragment shader. They begin facing a random direction, and their sex and number of legs is determined by a crude "DNA" consisting of two integer values in [0,9]. Females and males are currently identical except that females have a loop structure on top of their heads. They all move standard though, going forward unless encountering a wall or another crawler, when they turn on a preset value.

If they encounter another crawler, there is collision detection (on the bodies only, not the legs) to

handle one of three outcomes, unless the collision was between an unhatched crawler or a "dying" crawler. If they are of opposite sex, they will mate and produce a new egg which will later hatch. That egg consists of a random ordering of one piece of each parent's DNA (chosen at random). Currently, each crawler can only mate once. If the colliding crawlers are the same sex and have the same DNA sum, they just turn around and walk away from each other. If they have differing DNA sums, the one with the greater sum survives and the other dies. They can also die from old age, a preset value.

When the crawlers die, they "burn up" in an effect done via fragment shader. I achieved this effect by selecting a color (of three burn colors) based on the results of Perlin noise, seeded by the fragment's location in model space and a time variable. There is a primary color (50%) a secondary (40%), and a tertiary (10%). Since the noise is partially seeded by time, there is a "shimmering" or "burning" effect on the border of the "burn." The burn hole itself (discarded fragments) and the border are simply drawn via a sphere, with a radius determined by time, in model space.

References

Much of the vertex shader code, especially Phong lighting, is from your Example 27.

Noise function based on:
https://thebookofshaders.com/10/

Shader logging code based on:
http://www.hivestream.de/opengl-logging-and-debugging.html