Intermediate Computer Graphics

Final Report presented by GiT GüD Games

Requirements & Essentials

- Intermediate Computer Graphics
  - -> Modern GL v4.20 • Complete
  - -> Lighting • Complete
  - -> Texturing • Complete
  - -> Post-Process Color Correction • Complete
  - -> Particles • None
  - -> Bloom • Complete
  - -> Depth of Field • None
  - ->Tangent-Space Normal Mapping • Complete

Lighting

- Our plan for lighting the scene - The premise of the game
- The room in which our game is played will always be dimly lit using a single light source and an ambient light term.
- The character in our game is portrayed as a shadow in which any form of light source will kill them.
- The gameplay involves the passing of light sources moving through the scene and static sources with planned positions. The moving lights are on randomly generated paths through the area. The static sources have randomly generated grid spaces (e.g. 2x2 or 3x3)

Deferred Lighting

- We have a deferred rendering pipeline in place. Elaboration will follow in the *Texturing* segment.

Texturing

- Our plan for texturing is to use advanced texturing through deferred rendering. We will sample from albedo, normal, specular, emissive and depth maps.
- Color palette

Deferred Rendering

- Our deferred rendering pipeline includes deferred lights & shading. With the use of framebuffer objects with multiple render targets we achieve our desired outcome.
- Our geometry buffer (G-buffer) possesses the albedo map, normal map, specular map, emissive map and depth map for all rendered geometry. This helps us speed up per-pixel calculations.

Post-Process Color correction

- We are using a variety of color look-up tables. Some are being used practically in our game (fade to grayscale upon game over), and others are being applied on toggles for color correction effects.
- Algorithm for shader translation

Particles

- General overview of particles
- Geometry shader info
- Although our game has no particles implemented, we have practical applications of where they **could** have gone. For a simple effect, when our player uses the implemented dash mechanic, along with motion blur, we would leave a trail behind them. Next we have wall-mounted candles that light up around the room after a set amount of time. With this we would add smoke to the candles for a more fiery effect.

Post Processing effects: Bloom

- Our implementation of bloom was practical. The game world is a chapel like setting, where there are three large stained glass windows; each of which produces the effect of light extending from the borders of the windows, i.e. bloom. The stained glass looking textures will all appear to have sunlight shinning through or towards them, thus producing bloom.

Post Processing effects: Depth of Field

- General explanation of real life depth of field
- Description of depth of field algorithm
- Another topic that did not make it into the game but was still planned was depth of field. Our plan for DoF was going to be used in the game over of the player. Upon losing their last life, the scene fades to grayscale and displays a death message. Afterwards there is a scoreboard that appears. To help the player focus on the top five scores, we would make it so the background would become blurred thus making the game behind appear farther away.

Tangent-Space Normal mapping

- Algorithm description
- The topic we chose to implement was tangent-space normal mapping. This effect can be seen best on the floor of our scene, most evidently when a light passes over it. Our floor has a somewhat stone texture and this algorithm allows us to make this texture appear to have more depth; meaning the lines between each brick appear to be more depressed and thus darker. This effect makes the stones to look more individual and less as a whole unit.