Mandatory Assignment 2

Anders Fog Bunzel, 20112293
Mark Medum Bundgaard, 20112423
Mikkel Brun Jakobsen, 20114457

Friday 31st July, 2015

1 Representation of graphical elements

1.1 Blocks

Our world consists of 40 by 30 blocks of 20 by 20 pixels. Each block is represented as two triangles. Each vertex has a position and a color attribute. The color is chosen based on the block type. The buffer object containing block data is updated with (potentially new) data every frame. With this solution, deleting and creating new blocks is straightforward.

Block-data only needs to be updated interactively when it actually changes. With our current approach we update all blocks no matter what.

We could have used gl.bufferSubData.

Instead of representing the entire world in a single buffer, we could have partitioned it into multiple "chunks" and only update chunks that change.

1.2 Block-outline

We create a buffer containing 4 vertices once and for all. The block-outline is drawn using the LINE_LOOP mode. In order to draw the block-outline at the desired position, we use a uniform to offset it.

1.3 Stickman

We create a buffer containing 10 vertices once and for all. The Stickman is drawn using the LINES mode. In order to draw the Stickman at the desired position, we use a uniform to offset it.

2 Shaders

All our positions are in "block space". Each of our vertex shaders converts block-space to clip-space coordinates.

Our Stickman/block-outline fragment shaders simply outputs a ”hardcoded” color. The block fragment shader is augmented with the clickwave effect and a gradient based on a fragments distance to the center of the block it belongs to.

2.1 ClickWave

When the user places or destroys a block we start a ”click wave”.

We implemented the click wave by displacing vertices away from the outline of a circle.

We can configure the center/radius of the circle from Javascript using uniforms.

We pass on the result of the calculation from the vertex shader to the fragment shader and use it to slightly darken fragments close to the outline of the circle.

3 Browsers

Our solution runs on Chrome and Firefox. For some reason gl.lineWidth(5) does not seem to have any effect on 1 out of 3 of our machines.

4 Collision Detection

Note: Our collision detection is far from perfect.