Dylan Hartley and Nathan Regner

Bloxorz Recreation

For CS 464: Computer Graphics

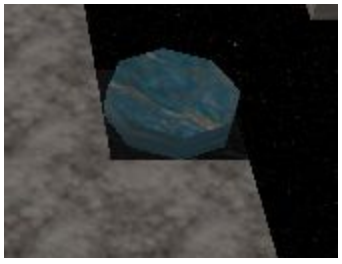
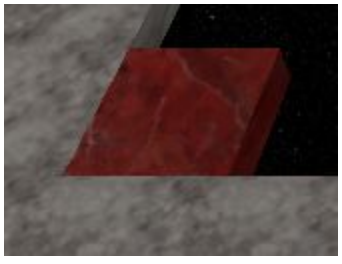
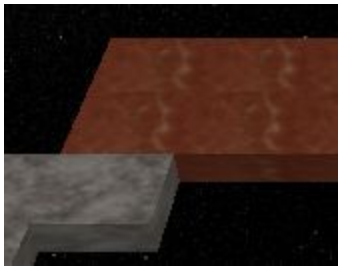
Professor: Steven Cutchin

User Manual

All bundled source files are included in the dist folder. From the dist folder, open index.html in the browser of your choice. We only tested with Chrome and Firefox, but it will likely work in others.

Once you have the webpage open, you can start playing. The block can be moved around the screen using the WASD keys. Your goal is to reach the escape hatch at the end of each level. There are a total of 5 levels.

To add additional challenge to the game, three additional tile types are included:

Button Tiles: Stepping on this tile will toggle other tiles within the level	**Toggle Tiles:** These tiles will appear or disappear when their corresponding button is pressed	**Drop Tiles:** These tiles will break if stood on vertically. You must roll across them sideways
		

Project Writeup

Development

1. Install Node.js
2. From the project root, run npm install. This will download all dependencies
3. From the project root, run npm start. This will start a Webpack dev server on http://localhost:8080. Source code changes will automatically be applied on save.

Architecture

Three.js
We used Three.js to do all of our rendering. It provides high-level abstractions for rendering 3d objects in a scene. Objects are stored in a tree structure, where the scene is the root. All objects can have zero or more children that can be dynamically added/removed as needed.

Typescript
We chose to use Typescript for all of our source code. Typescript provides some nice features that aren't yet supported in all browsers. It also provides compile-time type safety.

Webpack
We use webpack to transpile our Typescript sources into es5-compliant Javascript. In development mode, we also have Webpack setup to do live code reloading. Whenever you change a source file, the browser will be reloaded with the latest changes.

Source Files

- assets
    - background.jpg
    - block.png
    - button.png
    - buttontile.png
    - droptile.png
    - tile.png
    - weighted.png
- src
    - entities
        - block.ts - Creates the main block object.
        - entity.ts - Creates an interface for a 3D object
        - level.ts - Creates level objects
        - tiles.ts - Creates multiple tile objects

- animations.ts - Contains animation helper methods
- game.ts - Creates game class to initialize the game
- index.html - Webpage used to display the game
- levels.ts - Design for all the levels
- main.ts - Functionality for the game

Game logic

The Game class ties together most of the main game logic. Whenever a key is pressed, the game Game.onMove handler will be called with the direction the user wants to move. This direction is then applied to the block, which updates its mesh. The game then asks the block which tiles it is occupying. It performs a lookup on a 2d array of tiles, calling the onEntered method on each. This is how drop tiles determine if they need to break. Then, a check is done on each of the tiles to see if it is present. If not, the level is reset.

There is a special case hard-coded in the Game for the end tile. If the block enters this tile in the vertical orientation, the game will advance to the next level.

To deal with animations, we created an AnimationQueue class. All animations that block user input are entered into this queue. Before each move event is processed, the queue is checked. If there are unfinished animations present, the event is skipped. This prevents the user from moving while blocks and tiles are still animating. The level switching logic also drains this queue before switching to the next level or resetting.

Challenges

The inherent challenge of the game is to reach the empty tile placed within the level to move onto the next level. This can be challenging as the empty tile can only be accessed through the north or south orientation of the block which causes the user to have to think of a way to adjust the block to fit which at times can be challenging. Each type of tile creates a different type of challenge for the levels. For example, our weighted tiles (orange tiles) do not allow the user to touch them with the north and south of the block. The tile that adds the biggest challenge is our button tile which needs to be pressed with the north and of the block to then activate the toggle tile located elsewhere in the map that is needed to reach the end of the level. This toggle tile is loaded into the level when it is initialized but has the visibility set to false when the button tile is pressed the program then proceeds to find the toggle tile it is associated with inside the level and sets the visibility to true.

Fin