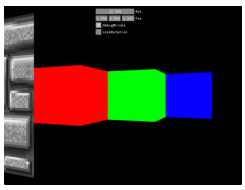
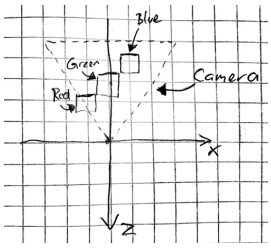
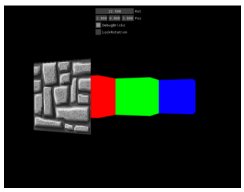
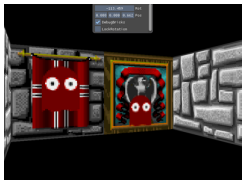
Game Programming – Assignment 2 / Exercise 6: 3D rendering

Learning objectives	**Learning objectives** • Use the view transform to position the camera in world coordinates • Create a simple FPS controller which moves the camera using AWSD and mouse input • Create a simple 3D model with texture mapping • Parse a level defined in JSON file using RapidJSON. You can discuss the tasks and possible solutions with your colleagues, but the implementation (code) is **individual**. **Handing in**: Create a zip-version of source-files, header-files and resources (CMakeFileLists.txt, .json and .png). We will build your project using CMake, so make sure it works before hand-in (Note: If you make sure to keep all files in the same directory it should work without you need to change the CMakeFileLists.txt). - Do not submit SimpleRenderEngineProject files. - Changes to SimpleRenderEngineProject are not allowed.

4-1
 
	First Person Controller Position and rotation
	 
	The goal here is to be able to move the camera by modifying the view transform. The initial scene looks like this:
	
	• FirstPersonController::update() should update the camera view transform based on the position and rotation variable. You can set the view transform using either camera->lookAt(...) or camera->setViewTransform(...).
	• Verify that your implementation works, by changing the values in the GUI interface.
	  ○ The position is world space position. Changing the x, z values should move the camera along the floor. Changing the y-value should move the camera up/down.
	  ○ Changing the rotation should rotate the camera around its current position. Positive rotations should result in counter-clockwise rotation around y axis.
	
	Solution: http://www.itu.dk/~mnob/wolf/wolf_1.html

4-2 	**First Person Controller Movement** • When on of A,W,S,D-keys is held down the movement should happened in relative to the current direction of the camera. The movement must be performed in FirstPersonController::update() and must use delta time (to make sure that the movement is independent of movement speed). ○ Hint: You need to distinguish between key down and key up events. • Mouse movement events should change the rotation when the relative x is changed. (See https://wiki.libsdl.org/SDL_MouseMotionEvent hint: "xrel") • Enable "mouse lock" by uncommenting the lines in the Wolf3D constructor Solution: http://www.itu.dk/~mnob/wolf/wolf_2.html

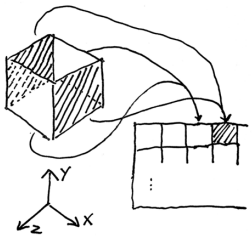
4-3



Create procedural wall with texture mapping

The structure of the input texture:
The input texture contains tiles of size 64x64 separated with 1 pixel outline.
The wall textures are indexed from upper left corner in the following order:
[0][0d] [1][1d] [2][2d] [3][3d] [4][4d] [5][5d] [6][6d] [7][7d]
[8][8d] [9][9d] …

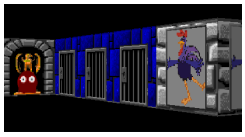
Where each tile exists in two versions: **normal** and **dark**. Normal textures should be used for faces in the xy-plane and dark tiles should be used for faces in the yz-plane.

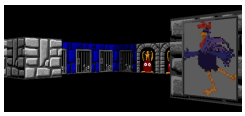


Modify the member function Wolf3D::addCube(). Currently it draws one quad (using two triangles) between -0.5 and 0.5 in the xy-plane (with z = 0.5).

- First create a cube (with a side length of 0.5) instead of just a single side. Note that the number of texture coordinates has to match the number of vertex positions. You don't have to create the top and bottom faces (only the 4 sides). Move around the cube to verify it looks correct from all sides.
- Take the parameters **x, z** into account and make sure that the cube is centered at **(x, 0.0, z)**. Look around – you should now be inside a small room with two wall segments on each side.
- Change the texture mapping, such that the correct tile is used as a texture, as described above (use the parameter **type** and make sure that the faces in the yz-plane are dark). Hints:
  - there is a lot of empty space in the top and right of the texture
  - the number 42 is not there because it is the "Answer to the Ultimate Question of Life, the Universe, and Everything" ☺
  - you may need the modulo (%) operator.

4-4

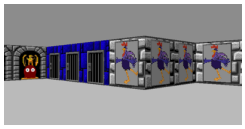




Parse JSON

The small room above is created in WorldMap::loadMap(), reimplement WorldMap::loadMap() such that the data of the world map object is loaded from the JSON file (instead of being hardcoded)

4-5



Create floor and ceiling (Optional)

Based on floor and ceil color (defined in the JSON-file and WorldMap) create one large quad for ceiling and another one for floor. See SRE sample project "04_spinning-primitives-tex.cpp" for an example (you might want to use "setColor" instead of "setTexture").