Version 0.02.53

A LOT has changed since I last updated this document. It’s been a month since I sent in my demo to Cloud Imperium and I’ve been super hard at work making Outland better. Below are the major changes that have happened as I’ve made so many the small ones are just too many to list.

* Finally fixed picking and got it working 100%. I have tied the collision detection for blocks into a brand new template class that keeps track of a reference to the nearest object so when it is done testing collisions I can immediately find out the closest reference from all objects that need to be tracked such as blocks, rooms, characters, or static objects. This also makes it incredibly easy to alter the position in case I need to change it (for instance build mode pushing out to the next block)
* I have taken many hours to get model import working correctly. Geometry is being passed in flawlessly and I finally got textures loading onto the models correctly as well. I need to keep an eye out because Lightwave exports collada files with an extra material for some reason so I have to manually go in and delete the “Default Dx-1” or something like that material. I’m quite happy that it is working so well and now need to focus on getting art work done so I can expand.
* I have fixed some errors in the build code that were revealed through testing. I’ve gone through and also converted the main build system over from using the clunky arrow key method to finally using the click and drag building approach. So far it works flawlessly but I’ve had to alter the land generation to produce flat land because I haven’t altered the build code to deal with more than 1 dimension in the y direction.
* I added a ghost building method for visually seeing what you are building when you are constructing/deconstructing rooms. This will show you exactly what your room will look like upon building it.
* I still need to add the math for the ray picking intersecting rooms since that will be more complex than a simple AABB. I’ll need to determine number of walls, wall orientation, corner status, and then finally determine if anything is hit when the ray passes through the room. I might be able to change the AABB formula to handle it, but I need to do some hard testing to ensure my theory is right on that.

Version 0.02.41?

* Added tabs to building window and added the secondary window.

Version 0.02.37-0.02.40?

Got a request from Cloud Imperium to submit a demo for my application. I decided to go against the standard rule and submit Outland as the demo so these version updates were frequent as I shifted between my laptop and main computer programming to get it to a playable state for them.

* Shifted GUI settings and transferred direct control of some methods (clicked and render notably) to the Button class that I’d been using to hold the textures for the buttons. Now the buttons will be responsible for maintaining the correct position of buttons and will be responsible for determining if a button has been clicked or not.
* Altered the Building GUI
  + Shifted the build and demolish buttons to the bottom of the window
  + Implemented several new buttons to determine new building types to build. So far, Command Center, Hallway, Living Quarters, and Bathrooms are available for building.
  + Added a selection button to the building GUI to visually show what building will be built. Clicking on another building will shift the selection button over top of the selected button to demonstrate the change.
* Resources GUI
  + Added more resources to the display of resources. Some may not stay. For now, I’m only using Steel as a building material.
* Optimization
  + Went through and converted some functions to a more efficient form. Also changed ALL constructer to use initialization lists instead of the less efficient allocation within body.
* Building:
  + As before, implemented different rooms. This included several new textures that are plugged into the engine.
  + Begin working on concept for placing doors. Initial testing proved VERY useful and seems to work without too much trouble!
* Tutorial
  + I went through and have created a messageBox class that can be used to display notifications to the player.
  + As a method to demonstrate this (and again to prepare the game for a demonstration), I designed a simple (skippable) tutorial that will display as soon as a new game is started. This will teach the user the basic controls and interactions they can perform.

Version 0.02.36

* Decided it would be a brilliant idea to try to get picking working again…
  + Added a debug line rendering functionality to the opengl renderer class and added calls to it in the update method of the graphics core.
  + Line debugging helped immensely. I now know that the positioning of the picking vector is incorrect.
* Started converting drawing to display lists
* Fixed another bug in the building code…hopefully the last one other than corners.
* Until opengl extensions are figured out for creating mipmaps, I’ve downgraded the block textures down to 128 to improve performance on the laptop. Main comp has no problem with the 512, but crappy vid card in laptop means horrible fps with that extreme of a texture repeated so much.
* Properly converted all classes to use initializers correctly. While core classes aren’t impacted much by this, the large object classes like block and room will experience a performance increase from it.

Version 0.02.35

* Resource Management:
  + Added functions to the manager class to affect player’s inventory
  + Fleshed out the playerInventory class since it was empty until now.
  + Added a resource manager instance to the gameplay core
  + Basic representation of power and oxygen on the resources screen. These are both represented as bars that will be green at full and slowly shift down to red as they get lower. I still need to write text on the resources page, but it’s looking good so far!
  + Readied other resources for implementation
* Removed a TON of warnings from type conversion
* Set base font to 16 instead of 14. Looks clearer at slightly larger font.

Version 0.02.34

* Fixed some of the building code to correctly merge. Before it was merging when buildings were placed diagonal to the rooms. Now it works as expected ☺
* Cleaned up some of the building code that was written unnecessarily. Refined some aspects so they were more efficient.
* Added a variable to the building method to determine whether the new building needed to be merged or not since the matrix wasn’t a 100% determining factor. Thinking back, I may need to alter the mergeRooms method to ensure it doesn’t copy rooms that are only diagonal to the room being built.

Version 0.02.33

* Added further switch cases to the rendering functions of rooms to plan for future expansion of textures and to get ready to implement more room choices.
* Fixed a logic error in the checkNearbyRooms function that solved the outer edges of rooms to not be recognized by the deletion function (minus a negative is positive…..)
* Set up visual distinction for power outages by tinting the textures red if the power is out in a room.
  + Changed render code for the buildings to accept a Boolean to indicate the power status so it only added a single Boolean value per room instead of per building block.
* Separated the loading of main menu textures from all other textures. Now main game textures will be loaded in a (not implemented yet) loading screen when the appropriate button is clicked. The loading screen will show the loading of textures, land generation, and model loading status.
* Found out that the merge isn’t deleting the previous blocks, it is keeping them in the Rooms array…need to fix that somehow….
  + Erasing of rooms seems to be fixed…Need to test on a larger scale with more than 2 merges occurring. Due to the way .erase() works, I may need to implement a dummy procedure to store the address of mergeRooms[0] since erasing an element from the array seems to screw up the incrementers for some reason…
* Need to fix the merge rooms function to ignore the corner pieces of the matrix since they won’t actually be connected. It will need to skip these over and refuse to merge if they are the only ones that are set to merge.

Version 0.02.32

* Code Rework on almost the entirety of the building code.
  + Added a Boolean value to the alterMergeBlock function to take both adding and deletion of rooms into account.
  + Fully fleshed out the AlterMergeBlock function to seemingly 100% working order. Haven’t found any bugs that are cropping up still with many many permutations of different buildings and removals.
  + Finished up the code rework on the Build routine to ensure that more advanced rooms could be built by merging with existing rooms.
  + Added code to merge the new rooms with existing rooms in the vector container. This includes a multiple added merge (a new room connects 2 or more rooms – they will all be joined into a singular room)
  + Improved the code rework by also writing a deletion routine to delete rooms with a given build square. So far it has worked flawlessly but more testing would be beneficial to insure no errors in the logic of the code.
* Began working on resource management
  + Resource management code has been started.
  + Power is now being calculated for the existing building types whenever a room is built. It is recalculated whenever the building is added to or deleted from.
  + Added enum for Resources to UnitAbstraction file
* Added in texture for the default floor grating
  + Fixed Texture cords for the floor and attempted to do so for the walls as well. Need to get the basic wall texture finished and imported to see how well that went. The floor texture needs tweaking still as it is not fully tileable, there is a small offset in on direction that is noticeable in the hallway sections.
* Added support for room textures to the renderer
* Separated the dirt block texture from GUI to its own Blocks folder in the texture directory and created a folder for room textures (where the floor grating is now).
* Separated the room wall textures from the floor and outer wall textures in code. I can now optimize the rendering code to choose the texture directly from the type instead of using a switch statement. I just need to make sure I load the textures for each room into their ENUM index number. This means I can change the switch cases into if statements for the rare case of a debug room or even better I can develop a debug texture and assign it an enum in the respective indices. Switch cases will still be needed for the outer walls since most of those will be the same and possible the ceiling code since I’m sure some of them will be the same between rooms.
* Added in an inner and outer wall texture for hallways as a test. It went rather well and I like the new look of the rooms with full textures on them ☺.
* Started work on ResourceManager
  + Added variable and functions to gameplay core to keep track of play mode and added calls within the main menu to initialize them.

Version 0.02.31

* Lots of changes to the internal building code. I pretty much gutted and remade it from the bottom up. Upon trying to get the building collision code working, I realized that I had forgotten to check rooms just outside of the building edges to see if already built rooms needed to be merged but none of its blocks were in the way of the new building.
  + Building code now uses a dynamic matrix which is one larger in both the x and z directions to check for the above case
  + Building code properly does type checking as well (needs extra building GUI to properly test dynamically)
  + Building code gutted and rebuilt using the new matrix operations for checking. Testing seemed to go successful so until further notice, this is marked as complete!
  + Added a precursor for changing other rooms – AlterMergeBlock. This block will change the blocks in the already allocated rooms to prepare for merging with the new blocks (IE takes the coordinates from the new build matrix for the block in question and knocks out any walls that need to be gone and changes the floor and ceiling accordingly.
  + Due to memory access issues with the new matrix implementation, the Alter blocks function had to be split into multiple to cope with the end boundaries of the matrix.
    - Need to fill in AlterMergeBlock function still but that should be the last step for the building mechanic to be fully functional!
  + Added 2 vectors to keep the list of rooms that intersect the build matrix ( for faster iteration in checking positions) and one to keep the list of merging rooms

Version 0.02.30

* This version is the version AFTER the final version for turning into the class. In short, picking algorithm was attempted and failed (possibly due to forgetting to reverse Y coordinates), Building system was demolished and rebuilt several times and eventually a build marker was implemented due to the picking problem, many new files/classes added, a few new textures added, menu system tweaked, AI system started to be written, Quaternions have been forsaken for the time being as I’ve got the camera working perfectly without them and I don’t need the full 6DOF that quaternions supply, and several optimizations taken into account and implemented. Everything following this is what has been done THIS version.
* Graphics Core:
  + Build system fixed from exploding wall bug that cropped up in the final release version because I swapped the rotation and translation commands for some reason.
  + Build system fully working.
    - The build system can now handle any square shape from large rooms to corridors 1 wide and even singular blocks.
    - The polygons have been tweaked to display correctly and tested thoroughly.
    - The render class has been split for optimizations and now contains switch statements that separate the possible wall code into sections.
    - The render class has also been prepared for eventual texturizing by having all similarly textured polygons grouped together under a switch statement that controls what texture will be applied to those walls based on the building type. All that needs to be done for those is adding the building types into the switch statement as cases and call the bind texture for the texture needed
* UI Core:
  + Fixed a negative camera movement issue and got the reverse cam move working properly. The coordinates of the camera are now being displayed properly and make logical sense for debugging purposes.

Massive panic between the latest version to get it done by deadline. Change log details missing for those revisions.



Version 0.02.21 (22)

* Screwed absolutely everything up
  + VS screwed absolutely everything up
  + Quaternions screwed absolutely everything up
  + Spent hours/days frustrated solving the stupid problems.
* Found out solutions:
  + Reverting the files from the SVN seemed to absolutely screw up Visual Studio by breaking the build files. Why it didn’t automatically clean and rebuild them, I don’t know.
  + Had an absolutely MASSIVE memory leak from the damned loader class. I forgot to delete the pointers for the file stream (not a big issue compared) and the buffers for the input and output (GIGANTIC FREAKING ISSUE!). Those are now deleted properly.
* Still haven’t repaired the quaternion class entirely, but I need to be able to reverse the quaternion matrix in order to render correctly which is a fairly intensive process
* Graphics Core:
  + Separated some of the code that runs on the engine startup to run instead when the player selects new game by adding a method to graphics core (startSingleGameplay()) and adding call to that when the menu button is clicked. Likewise I probably need to separate the deletion of the chunks and such to a class that ends the gameplay and returns to the menu.
* UI Core:
  + Insured that the camera rotation is always between 0 and 359 degrees.
  + Patched in call to graphics core to start loading the chunks when a new game is started.
  + Bounded the camera position to within the loaded chunks
* Engine Core:
  + Finally implemented my own working timer that will render the logic at 30 FPS at best case and render the graphics at 120 FPS at best case. This has taken some of the strain off of the CPU and is now running about 16% instead of 50% on my laptop in the main menu. The game is guaranteed to update at LEAST 24 FPS for both logic and graphics so even during the heavy lag times it still renders properly. Though this has the unintended side effect of potentially breaking the game logic. I need to experiment on my main PC to see the after effects. There might also be a problem with the clamping on the camera position.

Version 0.02.20

* Minor changes to Loader class

Version 0.02.19

* Player Core:
  + Added a building info class for storing the information of the different building types.
  + Added a room class to store all of the building blocks together into a common room.
  + Changed various small details throughout player class to extend functionality and get some things working
  + Worked quite a bit on the coding of the loading/saving system. I’ve almost got it ready for a trial run with a few chunks. I need to also edit the movement system to incorporate the change before testing so I can scroll through new chunks.
* UI Core:
  + Added several GUI specification classes to specify what GUI is active and the actions that need to be taken when it is displayed.
  + Created and implemented the GUI textures! Working perfectly!!!!!! Note: For colors to turn up correctly and not be tinted, glColor must be set to white (1,1,1)!
    - I may end up tweaking the system button since the keyboard tiles are being squished and it’s looking odd. However, every other button looks perfect.
    - I’ve only finished the Main GUI elements. The additional window and buttons inside still have to be made and added, but the coding isn’t finished for those yet. I’m working on that now.
  + Secondary GUI window coordinates worked out
  + Secondary GUI Textures started. Secondary window and close button done. Textures for each specific GUI still need to be done and assigned coordinates.
* Graphics Core:
  + Added a GUITextures container and updated the getTextures routine in the OpenGLRenderer to be able to retrieve from it.
  + Updated the batchBMPLoad to include loading of the GUI textures.
  + Started working on Viewport Frustrum culling algorithm.
  + Swapped a few member variables from private to protected and friended some classes to provide access between them.
  + Temporarily increased render distance to test textures
  + Added dirt texture with good results
  + Toyed around with landgen settings to get a more consistant and flat plain that wasn’t too flat….mostly unsuccessfully.
* Engine core:
  + Added a Vector4 template class for 4D equations like the frustrum culling.

Version 0.02.18

* Graphics Core:
  + Corrected the GUI implementation once again.
  + Cleaned up the Render code once I got the correct order figured out.
  + Added methods for rendering a single building room. Got single room building worked out and rendering correctly! This is large step to implementation to full build mode.
* UI Core:
  + Worked out the main GUI implementation.
  + Coordinates for the main GUI and buttons have been worked out and are ready to be replaced with their official textures. I might tweak the total size some more but all of the buttons and main GUI should update properly based on the size variables at the start of the GUI Render method.
  + Added a current GUI tracker variable to control what GUI windows are displayed. I think I’m going to go with a hierarchal GUI and have the main GUI own the other GUI windows. For whatever flag is active, it’ll call that feature’s render method and they will have further flags to determine the data displayed and such.
  + Changed the UserInput clicked method to properly route to either the GUI core or the Menu core based on the game state. Also made it slightly more efficient by only performing the calculations if the menu is active.
* Engine Core:
  + Added a UI enum to unit abstractions
* Gameplay Core:
  + Fleshed out the building class based on what information I obtained from the test of building singular rooms.

Version 0.02.17

* Graphics Core:
  + Started converting the debug chunk methods to do the proper job of loading and generating new chunks.
  + Added the landgen class to the graphics core class.
  + Filled out the chunk’s generate method to generate the block types for new chunks.
  + Added clauses in the render methods to ignore air type blocks until I can get the active/inactive culling added.
  + Successfully rendered 5^3 chunks with land generation algorithm!
  + Added a sample building renderer and toyed around with looks. It works fairly well right now at rendering simple rectangular rooms. I need to toy around with methods for non-rectangular and checks for the full building renderer.
  + Stripped graphics code out of each core due to the decoupling of the game logic and rendering processes. See next point….
  + Fucked up GUI….AGAIN. I think I’ve finally gotten it fixed AND gotten it switch to a working orthographic model! I’ve finally gotten it to accept the switch from perspective to ortho and back to draw the GUI on top of the screen. I need to recalculate the new coordinates however which will once again be based off of the resolution variables since it is now pixel-based.
* Began implementing time slicing for ALL cores. Laptop cannot keep up with graphics rendering otherwise.
* Corrected some of the engine wide warnings for data types
* Player Core:
  + Filled out the technology class with functions and variables
  + Filled out the technology management class
  + Added the 4 basic technologies to the startup list. More can easily be added later. Just need to add the type to the enum within Technology.h and add the starting values to the initializer in techmanage.cpp. If you’re loading a level, you’ll also have to load the new technologies from file as well.
  + Added a class for loading and saving of levels to the player core. Not filled in yet though.
* UI Core:
  + Found out that UI clips through blocks. Fixed it by readjusting the depth of it and refiguring out the coordinates for the in-game ui
* Engine Core:
  + Added an average FPS function to the Timer class and fixed up the code a bit so it’s cleaner.
  + Added a function to return the number of milliseconds the last frame took to be used in interpolation and timing for the main game loop.

Version 0.02.16

* Graphics Core:
  + Added method data into the LandGen class to generate the height of the land for new chunks.
* Player Core:
  + Might need to privatize the player’s data within the Player class. For now, leave public so we can determine functionality before privatizing.
  + Added a GameplayCore class to control all aspects of gameplay from the player to researching technology to the buildings.
* Engine Core:
  + Added the gameplay core to the main function for startup, update, and shutdown.

Version 0.02.15

* UI Core:
  + Changed the User Input click determination to the correct conversion from pixels to GUI coordinates. Need to look into the system API’s for getting the border width from Windows and other OS’s in the future to do this automatically if it is not fullscreen.
* Textures:
  + Converted all textures to .TGA format for simplicity’s sake. They should all now support alpha correctly and be able to show the backgrounds/game behind them.

Version 0.02.14

* UI Core:
  + Added a GUI class to handle the main game GUI elements when within the game now that the overlay is figured out.
  + Calculated a rough estimate for the creation of the GUI’s vertices. Need to test with a graphic to ensure proper texture coordinate implementation.
  + Fixed a missing #ifndef within the button class (How do I keep missing these?!)
  + Added calls to GUI’s startup within the UICore startup routine. Also added the GUI class definition to UICore.
  + Menu System coordinates have been mapped and textures are fully working. Only thing left for the menu to be fully operational is the algorithm used when clicking and comparing to the buttons on the menu. That shouldn’t be too hard since the positions are now relative across the resolutions. I can even optimize the algorithm a little bit by determining first whether it lies within the inner x boundary of buttons (the smaller buttons) I might change the button class to include the positions so that they can be iterated through quickly to check the variables.
    - Sidenote: All graphics that need an alpha transparency must NOT be BMP. TGA works. For now, I don’t plan on having a background image, but in the future it would be nice so the menu buttons will probably need converted to TGA format eventually.
  + Heavily coded the calculations from screen coordinates to local GUI coordinates inside of the UserInput class
  + Added extra variable and code to ensure the special case of a left click is responded to correctly. May change this code around since now that I think about it, I don’t want the UI’s constantly popping up and down from the quick keys. I may change it so that there is a keysUp vector list as well that compares what keys have been removed recently. This only pertains to UI controls. If for instance the jump key is held down, I want them to constantly jump up and down.

Version 0.02.13

* Engine Code:
  + Added functionality to the Timer class in preparation to convert all cores to be time-dependant. This will be a gradual change where the cores will take the current value of the FPS and change their actions based on how long the last frame took to render. Graphics will go largely unchanged since that’s the most important, but AI, input, and physics can all be simulated by protracting time out and splitting the workload between the full ideal 60FPS instead of every frame.
* Graphics Core:
  + Added the libnoise external library
  + Created a housing class for libnoise called LandGen. This will be the class responsible for calculating all of the new land generation.
    - I’ve included several modules to start tweaking values in and combining together. Some of them will NOT be used in the class demo as I want the starting planet to be rather soft and non-mountainous. I’ll add in some mountainous areas to be sure, but for the most part I want more of a plains area where beginners can build without worrying about terraforming.
    - Default values have been tweaked in the setup method. In final release, this method will be passed the type of the planet so it tweaks the noise libraries settings properly to form the kind of terrain intended.
* UI Core:
  + GUI drawing seems to be working perfectly now. The bounds have been figured out (though cannot be moved for some reason I’ve yet to figure out). Work on the menu will be done tonight and should be fairly quick to do. Once the menu is fully working as intended, all that will be left is the overlay GUI while playing and the controls for the GUI and menu to work. The controls may take some wonky math to get working right since I’ll have to convert from the bounds of the GUI space to screen coordinates and check the click event against them.

Version 0.02.12

* Changed far too much too many times to keep track of final changes.
  + End result: Sold Soul to Satan for Some Suitable Solutions to Solving issues (was trying to do that all with s’s…couldn’t). Anyway, I finally got the images to load properly after going through 4 different image loaders. Problem turned out to be the assignment of the GLuint texture wasn’t assigning to the global variable I had been testing with. Fixing that got the texture mapped to the cube. So, for now, I will be using SOIL’s library to load images since it’s easy and doesn’t require any more work checking it. However, it can’t do alpha channel on bmp apparently so all of my images that need alpha will need to be targa files (already tested and alpha is working with them).
  + Need to remove the extra Bitmap class on the next revision since I won’t be using it anymore.
* Changed the SVN to ignore the extra files that are constantly changed and created from VS. Now it should only update the core files and the other files will either be rebuilt upon opening the solution or rebuilt upon building.

Version 0.02.11

* Graphics Core:
  + Added functionality to the OpenGL renderer class to support changing between 3D and 2D rendering modes for the GUI.
  + Added an accessor function for the private texture banks of the OpenGL Renderer
* UI Core:
  + Added the function calls to the new rendering ready functions into the render function of Main Menu.
  + Filled out the render functions for Main Menu (hopefully correctly)
  + Changed around the Button initializers to work better
  + Added state check in UI Core’s update method to clarify what the ui core is supposed to do in each state.
  + Temporarily switched the initial assignment of gameState to STATE\_MAINMENU instead of the normal STATE\_INTRO.
* Engine Core:
  + Added framework for a timer class
  + Fixed a missing ifndef inside the Quaternion class (why it didn’t notify me of that, I have no clue)
* Player Core:
  + Added framework for technology and a technology manager

Version 0.02.10

* Graphics Core:
  + Forgot to close the file in Bitmap’s LoadBMP function – now closes at the end of the file. Also ensured that the loaded variable should be set correctly in all circumstances.
  + Switched Bitmap’s file opening system from the potentially unsafe fopen to fopen\_s.
  + Wrote methods in OpenGLRenderer to bind the textures for blocks into OpenGL code. This is implemented in singular and batch methods.
  + Added loading of BMPs for menu to the startup of graphic core.
* UI Core:
  + Added the main menu to the list of the UI Core variables so testing can begin with the menu.
  + Added a texture pointer to the Button class so the buttons and menu has direct access to the texture.
  + Added a function to the menu class to set the texture pointers for the buttons and title.
  + Refined the render procedure for the menu class, working towards full menu UI testing

Version 0.02.01

* Graphics Core:
  + Added Bitmap.h and Bitmap.cpp to handle the loading and unloading of bitmap images for the game. These will be directly fed into the OpenGL Renderer to create the OpenGL textures for each object.
  + Changed minor features within Block class
  + Added a LoadBMPImages function to the OpenGL Renderer to handle the creation of textures.
* Engine Core:
  + Added more functionality to the Quaternion class. Added the matrix to quaternion mathematics courtesy of gamasutra article.
  + Added basic types to the Unit Abstraction class to deal with functions that cannot take a signed/unsigned type (notably fopen() in the bitmap class methods)