Meeting minutes 2.4.11

Architecture

* Many game play engines to choose from
* Several already made engines at our resource
	+ Use pre-made engines if possible
	+ If we decide to make our own, we could be facing over 1000 hours of programming
* Some example of engines are the unreal engine, unity engine and the Tort game engine advance
	+ The Tort advance engine can hold about 200 average users connected at one time
	+ It has the ability to easily add servers if the need arises
* Different servers:
	+ Game server
		- Where all players will connect to the game
	+ Peer-to-peer
		- While a good amount of data goes into peer-to-peer, a good chunk will still go through the Game server
		- We may need to go with a peer-to-peer-to-client server set up. This is where the peer gets the data from the client then becomes a node itself for other peers.
	+ Web-server
		- This is generally only used for signing into the game
		- Voice-chat can be routed through the web server due to its largely inactive span so the extra processing power can be put to good use

Visual and Navigation

* The students use zooming features if they are able to see
	+ Need to accommodate this feature into the game
* Learn to navigate quickly and easily
	+ Reminders: How they should be implemented and hot to explain each reminder in speech only
* Go through each menu both visually and vocally
	+ What word and their order makes a large difference in the final design
	+ Shorter messages are better. About 3 seconds until the message is forgotten
* The WHERE AM I PROBLEM
	+ This asks where you are at on the screen and how to say where you are currently located
* For status updates
	+ Sonification may be useful for this part. The use of a sound to represent a status instead of a word
	+ Different sound effects represent different statuses like remaining health
* Background music
	+ Needs to be soft and always behind the scene
	+ No looping or incredibly repetitive sounds
	+ If you must loop, the longer the sound files the better
* Balancing messages and chat
	+ Possible stopping chat for notice to be played
	+ Background noise behind the chat
		- Possible soft noise or a hand shaking kind of deal
* Audio and how it relates to the game is a large part in the project. Random sounds could detract from the game and make it harder to play

Educational Purposes

* Possible settings that would cause the player to enter a world where every X events will trigger a message about what is going on behind the scenes
* Make sure the game is fun to play
* Students should learn about the game and the processes that make the game possible
* Students could write Macros for the game in HOPS
	+ An example would be to write a path finding algorithm in the game
* Possible programming points are Macros, Levels, and limited item generation
	+ All mods must go through a submission process to allow them to be added into the game
* Modding of levels for personal use
	+ Character micros/moves and personal levels can be stored into a file on their own computers
	+ Their character files should never be stored on their computers. If they are, the users can then modify their characters
* Dungeon files
	+ Need a way for HOPS to work with the dungeon files
	+ Scripting in the game i.e. following paths and macros differ from actual dungeons
	+ Dungeons need to get approved before they are added into the game. Scripting should not have to get approved
* Possible learning dungeons can be implemented into the game by teachers
	+ In the place of monster attacks a quiz like system can be added
	+ The use of a party system for group work or quizzes
	+ Teachers would be in charge of designing these dungeons to fit into their classes

Miscellaneous points

* Magic: Magic may be implemented into the game without worry of some student’s religious or personal opinions that may cause them to not wish to take part in the class
* Possible to work in an economy/market place into the game
	+ Might want rules on how to behave in this setting
* Possible setting of the game could be around a school with a storyline revolving around this
* Some of the better games have different types of “quests” for their characters to explore
	+ Variety is essential for these games
	+ Quickly made dungeons for testing are a plus for designing these games
* Talent trees
	+ Different from player type to player type
	+ Allow for easy customization of characters
* less garbage items in the game

Things for next meeting

* Work on level 0
	+ Basic understand of controls
	+ Scripted tutorial for level zero
* Dungeon API setup
* Small parts of the game should be tested for usability as the engine progresses
* Writing lessons either inside the game our outside of the game