# LameGame

Aaron Gonzales, Robert Nicholson, Weston Ortiz, Paige Romero, Hans Weeks

November 7, 2014

# 1 LameGame (Final name undecided) Specification

# 2 Brief Description of LameGame

LameGame is a first person shooter with spaceships. In LameGame, a player will have a view of the spaceship from above and back, watching it move around as it goes through space. The game environment will have various obstacles, notably asteroids that can destroy a ship if not avoided or destroyed by the user. In multiplayer mode, up to four archrivals can work out their issues by attacking each other or tricking a user into hitting an asteroid.

Littered throughout the game world, there will be various power-ups, weaponry, and health refreshers that players may obtain by flying through them. Some powerups may not be so powerful. Some may be very powerful indeed.

# 3 Scope

The scope of LameGame is mostly the five developers, our TA, Torin, and our instructor, Joel - unless for some reason the general public finds a major interest in our amazing game, LameGame. It is meant to be recreational for all except for the developers.

# 4 Definitions

- Server: A machine that serves as a centralized point of entry for users during multiplayer access. It can store user data and gameplay information, calculate physics and interactions, and handle placement of the aformentioned all-powerful powerups. The server's physical location has not been determined, though a machine in FEC may suffice for our purposes. Users can connect to the server through the Internet, because it's 2014. Basic packet verification should be done to ensure only correct data is being sent to the server.
- UI : The user interface for LameGame. This serves as the primary (and only) way of interacting with the program. Options to reload levels, start games, connect to the server, pull user data from the server, close the game, and so on will be present through the UI. The gui most likely will be done in OpenGL unless swing/java fx provide a better interface.
- Player : Person who is playing the game to the best or worst of their abilities, in single-player or mulitplayer mode. Let no designation be made between 'player' and 'user'.
- Input: as a subcomponent of the UI, input will be handled through a joystick (most likely wired xbox controller) and/or keyboard and mouse.
- Graphics: LameGame will be highly visual and beautiful. Some (members of the dev team) may say it looks like Rothko or another famous artist who didn't work in concrete shapes or forms. LameGame's graphics will be made using Java OpenGL (JOGL) via the Lightweight Java Game Library (lwjgl). Developers will attempt to avoid creating new acronyms. Graphics will include meshes, lighting, textures, immersion, moveable cameras, and "realistic" movement of the objects in space, including the player's ship and all other things. This will be a large chunk of work.
- Physics: We are looking to implement somewhat believable space physics with inertia playing a part in player movement/gameplay. We are in the process of looking at either implementing basics physics on our own or integrating bullet engine through JBullet. We also want to have asteroids moving through the game area and have the ability to explode, on contact with projectile or enemy ship.
- World: The game's current world or map in which players may play. The world will be comprised of several decorative objects:
  - the world will have a starry background that is everpresent, like the stars themselves.
  - The world will have far-away planets to provide a sense of place. These planets will hopefully move a little bit, because everything is moving in space.

  - the world will have 3d objects in it, primarily asteroids unless we have time to add more objects (e.g., space mines, space pirates, space ninjas, space princesses, space beach balls, space aliens, floating software specifications that devour you in one 'hit', evil Computer Science instructors that think you have nothing else to do but work on their class).

- Testing: A thing that young developers seldom do correctly but is crucial to the success of a larger project. Testing within LameGame will include three subcomponents:

  - Unit tests - these shall be written for each class or collection of classes that inherit each other.
  - Integration test - these shall be performed to ensure that the disparate parts are hacked together in a way that works and doesn't break some other component in the process. It is the author's suspicion that this will be challenging.
  - 'Daily' builds - after the first semi-working version is complete, one member will be in charge of creating a 'daily build' of LameGame. This daily build will be a final bugfinding and integration testing measure . This daily build may not in fact be daily, but may happen many times per day as the deadline approaches. Also, the authors do not expect that compiling LameGame will take that long - it's not Windows Vista, after all.

- Version control / code reviews: Git through Github. We students will be using Git/Github as a DVCS, implementing a standard feature-and-bugfix branch workflow because we all want to do things that future employers may like. A 'dev' branch and a 'master' branch will be the primary lines of development, with merges into master being handeld by one crazy group member. We will make extensive use of feature branches and pull requests through github to ensure that code standards, code reviews, and tests are implemented correctly.

- Game interaction: Any interaction between a Player and the world. This includes events such as picking up an incredible powerup or firing a space plasma rocket grenade laser at an archnemesis.

- Documentation: Refers to both JavaDoc, specifications documents like this one, notes, and smoke signals that relay information to each other or to an end user/player. This includes technical and non-technical docs.

# 5 General breakdown of responsibilities

Note that distribution of labor may change slightly as the project's completion waxes and wanes, but for right now, our breakdown is as follows:

## 5.1 Aaron Gonzales

- "Leader" or something like it.
- Tracking development process and assigning release dates
- Testing
- Daily builds
- VC/ Code reviews
- Assist with server and physics implementations
- OO design and layout for subcomponents
- Performance testing
- Documentation
- Pedantic musings

## 5.2 Robert (Dan) Nicholson

- Implementing graphics (not as simple as it may sound!)
- Implementing Audio / OpenAL
- Moral support

### 5.3 Weston Ortiz

- General physics and game interaction/logic
- Helping with the server and communication
- Collisions / network related logic for keeping collisions correct on client side, while server does most of the collision checking (hits/pickups)
- '2nd lead' assisting with integration if needed

### 5.4 Paige Romero

- Graphics implementation
- Models, textures, lighting
- UI/UX for game

### 5.5 Hans Weeks

- Server and client communication
- Containers for client game-state update
- Game physics