Project Lyra WEB1: Real Time Strategy

Contents

Project Team

- ➢ Mark Giles
- ➢ Corinna Huffaker
- ➢ Runa Trinh

Introduction

The Lyra game is loosely based on the 1984 Video Game “Spy vs Spy”. The object of that game was to collect various secret items in a briefcase and exit the building through a door to the airport before the opposing player, or before the timer runs out. (reference: https://en.wikipedia.org/wiki/Spy_vs._Spy_(1984_video_game))
Project Lyra will be making a 2D HTML5 web based game with RTS aspects. The goal of the game is to provide an experience where the player attempts to locate randomly placed objectives on the map while evading or blocking AI based challenges. A time limit will be used to create additional difficulty. As a team we have limited experience with game development. The team is also challenged geographically by having team members in Cincinnati, Japan, and Seattle. All team members anticipate completing the OSU curriculum this quarter with Project Lyra being our final capstone project.

Client Requirements

The instructor, Benjamin Brewster, will be considered the client. The project will be designed to meet the requirements defined by “WEB1: RTS” document. Including the following game elements:

- Single player
- A user's progress must be able to be saved, such that the user can leave and come back to resume the game. You can use cookies or a server-based DB for this
- Your game must allow individual "units" to be constructed that serve some purpose in a 2D area (combat, harvesting, painting coverage, etc.). Note: The escape pod repair described below is intended to meet this requirement.
- There must be a computer/AI opponent that resists and plays counter to your efforts. This AI must have 2 distinct settings - an easier mode, and a harder mode. The AI must be able to win, which causes the game to end, requiring a restart. Note: Bandits and slime working in opposition to the player are intended to meet this requirement.

- An individual game, screen, or board must be winnable, with reasonable to serious effort/skill, within 10 minutes. A new player should not be able to beat the game. Consider providing multiple screens or boards as the player progresses.
- If your game has units attacking or otherwise affecting something, appropriate animations showing projectiles, beams, healing, etc. must be shown to indicate what's happening.

The following platform and technical elements are also required:

- Your project must be playable using the Google Chrome web browser.
- Your project must be hosted somewhere for me to test for grading purposes, though you will be required to submit source code. It does not have to be served from an OSU web server, but this would be preferred.
- You may use game libraries, frameworks, and toolkits such as Melon JS, Tiled, Texture Packer, Sigma.js, vis.js, D3. You may also use more general purpose libraries such as Node.js, bootstrap, React, Angular, and jQuery, if you like.
- Do not use Adobe's Flash player or any other encapsulated app.
- Do not use Javascript-style "alert()" pop-up boxes to relate information to the player. Your user interface should display all information without resorting to asking the web browser to pop-up a dialog.
- Data about the individual enemies and units must be loaded from separate files on the web server (one enemy or unit per file).

User Experience

Under a time sensitive environment, the user will need to quickly search for resources and manage them in order to make their escape. At the same time they will need to either actively avoid or engage the enemy in order to take back stolen resources. It’s a game that will encourage the user to engage in quick decision making and test their resource management skills. The user can also complete additional objectives for extra points that can either increase or decrease the difficulty of the game based on the user’s skill. The game is designed with replayability in mind as the user should learn to get better at each playthrough coupled with the fact that there are randomized elements to the game.

Initial Game Plan

Story

After a successful mission retrieving the rare pink diamond, the Lyra ship and its crew make their journey back to home base. Unfortunately, the ship has been intercepted by space pirates and in the process irrecoverably damaging the ship. As security android OS-U, you must find the parts to repair your escape pod before time is up and deliver your mission report to HQ. Secondary objectives are to rescue the rest of the crew, help them escape and if possible bring back the pink diamond.

int o2level(0 game over/failure, 0 < x active game)
bool escaped(0 completed escape pod not used, 1 completed escape pod used)
bool pinkDiamond(0 pink Diamond not in crew inventory, 1 pink Diamond in crew inventory)
bool allCrewFound(0 false, 1 true)
bool allCrewEscape(0 false, 1 all found crew members have escaped)
bool stolenDiamond(0 pink Diamond not taken offship by pirates, 1 pink Diamond taken offship by pirates)
bool crewStuck(0 false, 1 all found crew is stuck)

Game Flow

The user must work against the clock before oxygen runs out.

1. Find an escape pod.
2. Repair the escape pod by finding the 3 items needed
3. Defend the escape pod/self from the pirates.
   - ❏ Trap them in rooms.
   - ❏ Run away.
   - ❏ Knock out with sleeping gas.
   - ❏ Take back scavenged items and/or steal stun ray.
4. Optionally: Find and wake the other 2 crewmen up.
   - Pros: Help to search for things faster.
     Higher Score.
   - Cons: Quicker oxygen depletion.
     More to manage.
5. Optionally: Find the Pink Diamond.
   - Pros: Higher Score.

Cons: Takes more Time.

6. Escape in a pod before the oxygen runs out.

Game End Conditions

Major Victory

- ❏ Repair a Pod and Escape
- ❏ Retrieve the Pink Diamond
- ❏ Escape with all the Crew

Minor Victory

- ❏ Repair a Pod and Escape.
- ❏ Do Not Retrieve the Pink Diamond Or Pirates Escape with the Pink Diamond
- ❏ Crew members lost.

Failure

- ❏ User does not escape before the Timer(O2 Levels) hit 0.

  Or
- ❏ All active crewmembers get trapped in slime.

  Or
- ❏ Pirates take the pink diamond back to their ship.

Strategy Choices & Game Elements

Crew Members

- ❏ Located in the crew quarters in sleeping pod “containers”
- ❏ Must be released by another crew member.
- ❏ The number of “awake” crew increases the rate of oxygen depletion.
- ❏ Actions: Move, Pick Up, Drop
- ❏ Can’t move if stuck in slime.

Menu

- ❏ Login/Create Account Page
- ❏ New Game
- ❏ Load Game
  - ❏ Loads a previously saved game
  - ❏ Can only be selected when there is a saved game
- ❏ Game Settings
- ❏ Gameplay instructions

Landing Page

Create Account Page

Main Menu

Within Game

- ❏ Must hit the Menu/Esc button
- ❏ Save Game
    - ❏ Saves the current state of the game

User Interface

Controls

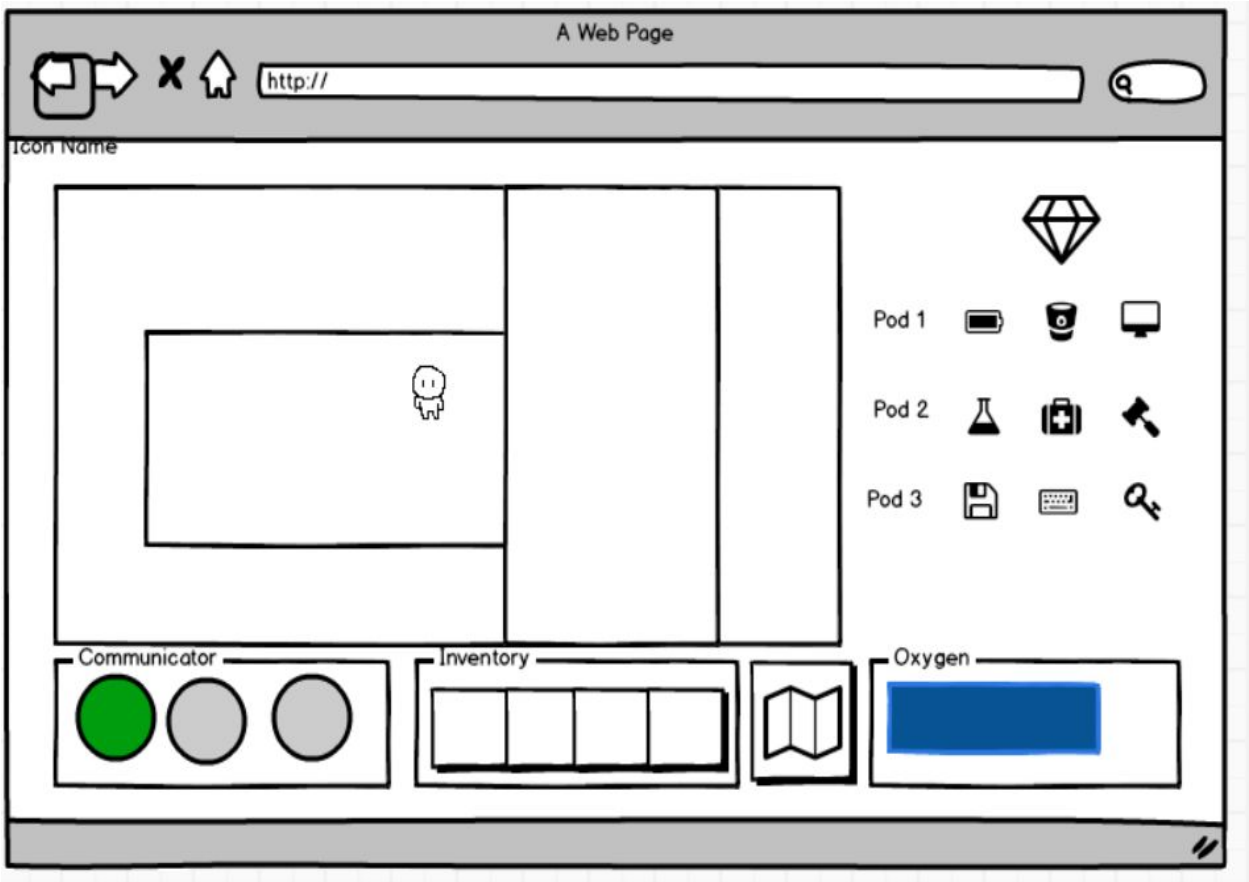
- ❏ Movement
    - ❏ Up, Down, Left, Right Mapped to WASD/Directional Buttons
- ❏ Interact/Cancel
    - ❏ Mouse
- ❏ Menu
    - ❏ Escape or M Button

- ❏ Save Game
- ❏ Load Game
- ❏ New Game

Heads Up Display

- ❏ Timer
  - ❏ Represents the amount of O2 in the ship.
- ❏ Objectives
  - ❏ Pink Diamond
  - ❏ List of Escape Pods and Required Items.
  - ❏ Found → Grayed Out
  - ❏ Required Items represented as Icons
- ❏ Map
  - ❏ Shows the entire map.
  - ❏ Blinking dots to show crew member locations.
- ❏ Communicator to each Crewman
  - ❏ When clicked, it switches control.
  - ❏ Unfound→ Gray
  - ❏ Awaiting Orders → Question Mark
  - ❏ Green → Destination is Set
  - ❏ Yellow → Stuck in Slime
  - ❏ Red → K.O.
- ❏ Inventory
  - ❏ 4 Slots
  - ❏ Small → 1 space
    - ❏ Big → 2 spaces

Interface Prototype



Rooms & Containers

Map is a giant spaceship with a predetermined layout. The user has a randomly generated start position.The user only sees one room at a time(?) and it's a top down view. Each room has "containers/searchable areas" which have predetermined locations. They also have doors which can be opened or closed at any time.

Container size can hold multiples of the same resource and vary in the number of resources they can hold.

Engine Room

Containers: Locker, Box

Command Center

Containers: Desk, Computer

Crew Quarters

Containers: Box, Sleep Pod

Workshop

Containers: Tool Shed, Locker, Box

Medbay

Containers: Locker, Bed, Box

Docking Bay

Containers: Box, Box, Box

- ❏ Pirate AI is generated in this room.

Hall

Containers: Slime Suppressant

Rec Center

Containers: Box, Locker, Locker

Cafeteria

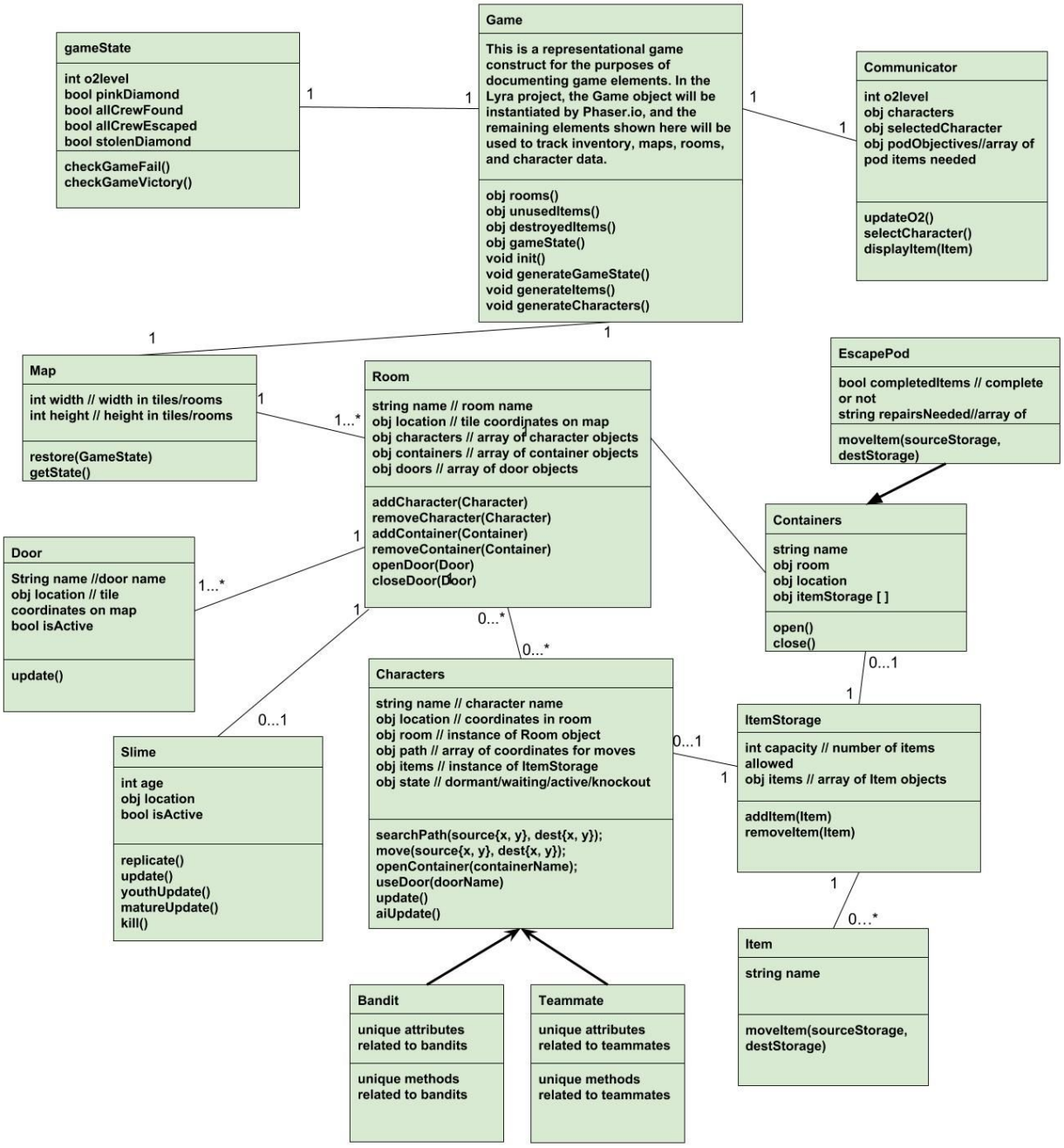
Containers: Fridge, Table

Escape Module x 4

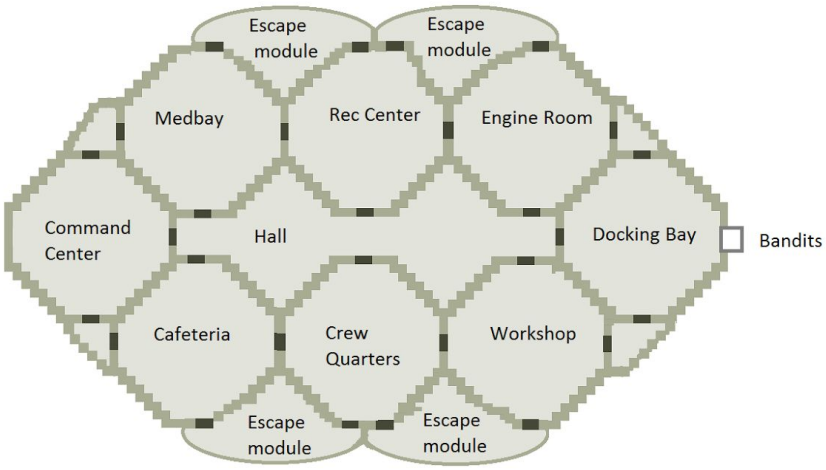
Special Container: Escape Pods

- ❏ Status: Damaged→ Repaired
- ❏ Required 3 randomized resources for status change.
- ❏ User drags and drops resource on the pod to “repair”.

Game Object Model Prototype



Easy Mode Map Prototype



Resources

Resources are randomly generated in containers across various rooms. Each container can contain multiple raw materials.

Resources: Fuel, Battery, Spark Plug,Water, Oxygen Tank,Circuit Board, Wires, Valves, Rods, Sensors

Difficulty

Easy

- ❏ Smaller Map Size
- ❏ 3 Crew Members
- ❏ 2 Bandits
- ❏ 3 Slime

Hard

- ❏ Bigger Map Size
- ❏ 3 Crew Members
- ❏ 4 Bandits

- ❏ 6 Slime
- ❏ More Aggressive AI

Enemy Behavior

Pirate

- ❏ 2 pirate enemies get generated in the docking bay for easy and 4 in hard.
- ❏ Starting Inventory: Stun Gun
- ❏ There is a cool down time of “30 secs”(?) before the pirates can “leave”(generated into the dock).
    a. Sees if there are unchecked containers in the room.
    b. Finds the closest unchecked container.
    c. If there is a resource it takes it.
    d. Else it finds the next closest container.
- ❏ If AI encounters User, it will either:
    a. Ignore(Easy Mode?)
    b. Try to attack the User rendering them unable to move for two seconds and making them drop a resource.
        i. Steals a resource if the user has any.
            1. Easy-(Takes only one?)
            2. Hard-(Takes everything from the User?)
        ii. Leaves the room.
- ❏ When the AI has found the Pink Diamond and it has all 3 resources needed for a pod, all AI will head to the pod.

Slime

- ❏ Slime is randomly generated into rooms at different time intervals.
- ❏ It replicates around the room.
- ❏ Each slime increases rate of oxygen depletion.
- ❏ If it collides with the user or user walks on slime, then the user gets “stuck”.
- ❏ Destroyed by a slime suppressant.

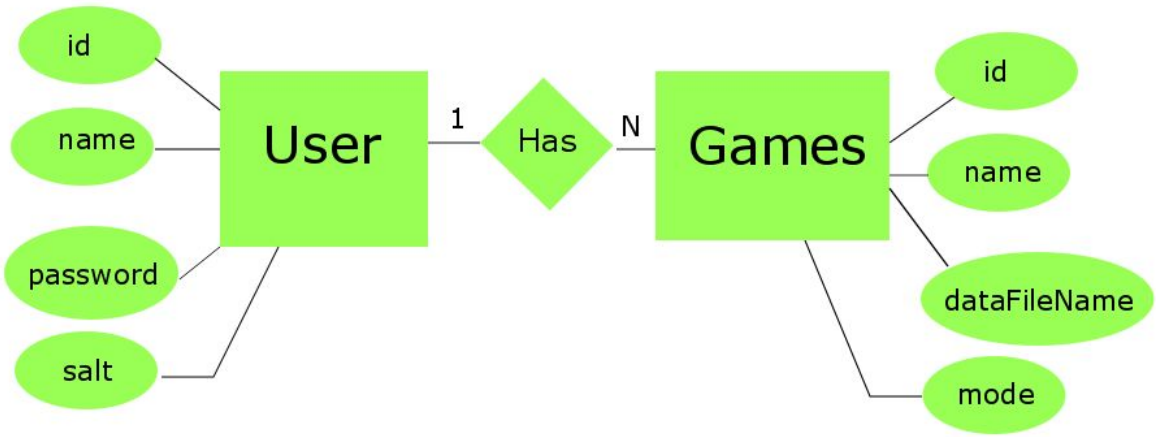
Architecture

Administration Framework

The administration framework will be provided by a server side component using PHP and MySql. The basic structure of the framework requires authentication with database-stored user

credentials. This will allow the game to identify players before beginning, and enables features such as saved game data and multiplayer components. The following entity relationship diagram illustrates how users will relate to saved game data:

Database ER Drawing



Preserving and Restoring the Game State

The attribute “dataFileName” stored in the MySql database table “Games” will be stored as a JSON object, and retrievable through JavaScript for loading and saving game stage information.

Source Management

The primary source of code management will be the GitHub repository located via git at:

https://github.com/mgiles0113/lyra.git

The master branch will contain the most up-to-date stable release version of all source code, and unreviewed code will be stored in branches with names that correspond to the Game Element name in the chart below followed by the version number. For instance, development code related to the Web Framework element that currently has three team-approved revisions will be located on the branch called “WebFramework_V4”. This will allow team development to continue using a central source without interruption. Once the code is reviewed and approved by the team, the branch can be merged to master, overriding previous versions. The source code includes two PHP parameter files that allow for easy portability to various environments:

- ❏ SYSTEM_PARAMETERS.php
- ❏ GAME_PARAMETERS.php

These files allow various development environments to run the same source code with unique system settings. These files will be included in the git .ignore file and should not be added when committing files to the GitHub repository.

Application Requirements

Server Operating System: 14.04
Web Server: Apache 2.4.7
Dynamic Web Content: PHP 5.6.28
Data Model Database: MySql 5.7.17
Data Model File Storage: Object storage folder, files stored in JSON format
Game Mechanics: Phaser.io 2.6.2
Styling: CSS3 with Bootstrap 3.3.7
Interactivity: JavaScript including JQuery 3.1.1

Development Environment

The team development environment is located on Cloud 9 at:

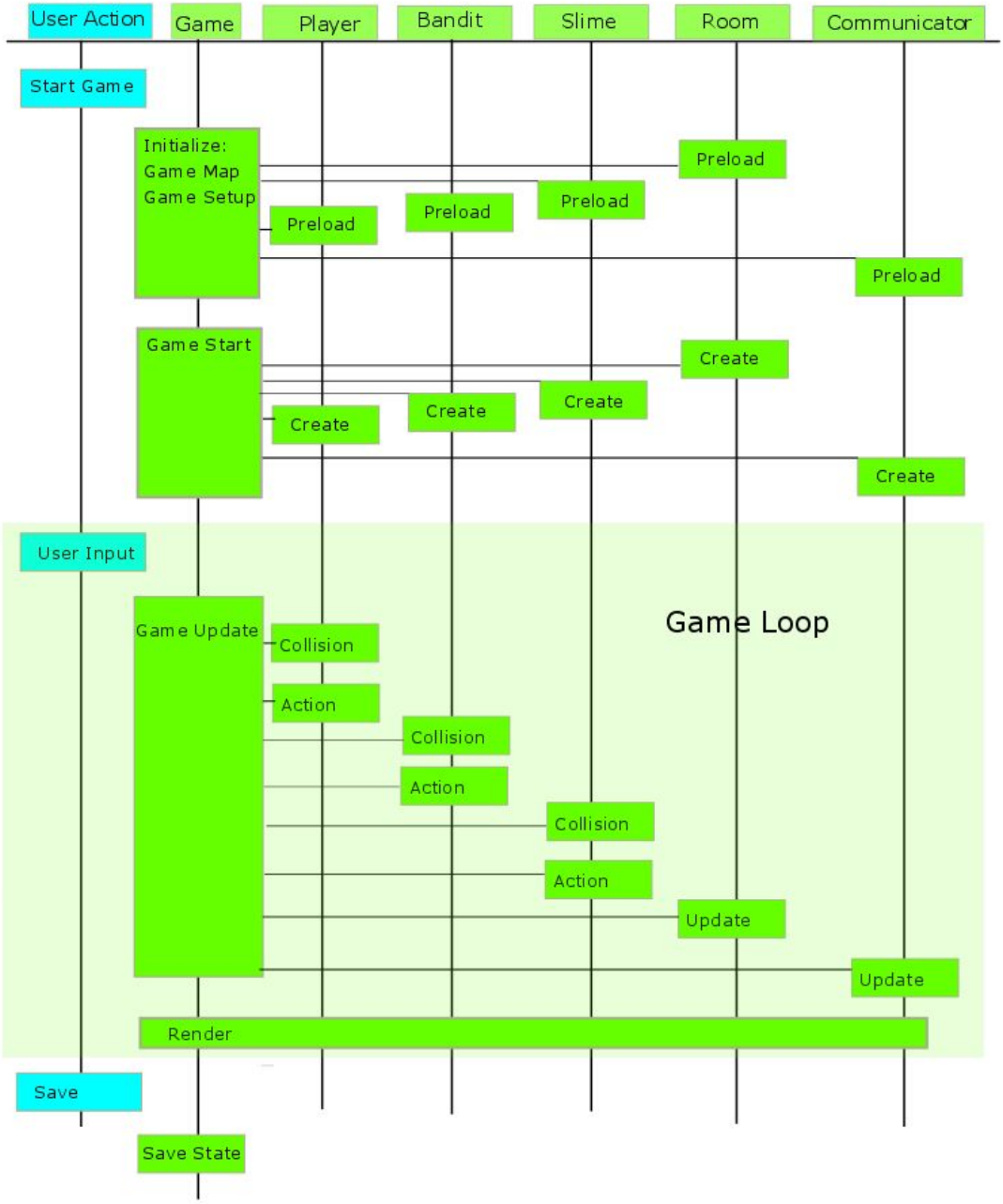
https://ide.c9.io/mgiles0113/cs467_lyra_development

All team members have read/write access to the repository and can use it for code testing, collaboration, and other demonstrations. However, refer to source management above for details regarding using any other personal development environment seamlessly.

Production Environment

Production environment will be located on a Ubuntu Server 14.04 hosted by GoDaddy at 192.169.167.218. Each team member will have SSH access to the web root directory for the application and will be able to edit files appropriately. However, the preferred method for using the production environment will be to use Git pull on the master branch after each release version is approved and published.

Game Sequence Diagram



Elements from Phaser Library

We are expecting to use at least the following resources from the Phaser.io library:

- ❏ Manage game state (preload, create, listener, update (overloaded in objects))
- ❏ Manage game loop
- ❏ Manage rendering to <canvas> element
- ❏ Physics engine

Phaser.io source code is available at: https://github.com/photonstorm/phaser/tree/v2.6.2
Phaser.io is released under the MIT License.

Art Resources

- ❏ Free or Free for Educational use licensed art found on internet.
- ❏ Original artwork by team members.
- ❏ Maps created using the "Tiled" program at http://www.mapeditor.org/.
- ❏ Purchased art where a team member holds the license to the work.

Plan

Documentation Schedule

Week	Due Date 11:59 PST	Report	Team Member submitting Report	Report content
3	27-Jan	Progress Report 3	Corinna	Individual videos
4	3-Feb	Progress Report 4	Mark	Individual videos
5	10-Feb	Progress Report 5	Runa	Individual videos
6	17-Feb	Mid point Project Check	Corinna	Corinna/Runa: Instructions Mark: Deployment
7	24-Feb	Progress Report 7	Mark	Individual videos
8	3-Mar	Progress Report 8	Runa	Individual videos
9	10-Mar	Progress Report 9	Corinna	Individual videos
10	17-Mar	Final Report / Demo	Mark	Corinna/Runa: Instructions Mark: Deployment All: Report

Major Functionality Development Schedule

Game Elements	Owner	3	4	5	6 Mid-point check	7	8	9	10 Final Report
Setup Development Environment	All	Access to C9, Github Repository, Organize File Structure							
Setup Production Environment	Mark	Build application requirements, user access	Practice transfer, System params	Pull published revision	Pull published revision, Deploy for grading	Pull published revision	Pull published revision	Pull Published revision	Deploy for grading
Create Database	Mark	Create Account/ Login Table init	Game table, sqli interface	Error checking	Bug Fixes	Multiplayer Development			
Web Framework	Mark	Initial game state, menu architecture	receive data from (JS Ajax), save	Retrieve data Restore Game State	Bug Fixes	Multiplayer Development	Screen Transitions/ Graphics		
Basic Game Framework	Corinna	Basic game loop	Save button, User input	Load new game	Bug Fixes, Load from previous	Communicator - status	Actions / notifications		
Canvas Map Interaction with game framework	Runa Corinna	Load basic map with rooms	Place collidable objects/containers on map	Place items in containers on map	Bug Fixes	Add multiple team members (sleep state)	Zoom	art	Bug Fixes
Room Object	Mark	Basic, dimensions	Add doors	Update, collision	Bug Fixes	Escape pod		art	Bug Fixes
Player Object	Corinna	Basic	Cursor control/ auto control	Select destination	Bug Fixes	Switch players	Knock- out	art	Bug Fixes

Container Object	Runa	Initialize attributes, placement	Dimensions	Update, collision (open/close)	Bug Fixes	Move locations		art	Bug Fixes
Item Object	Corinna		Basic template	Additional items	Additional items	Additional items	art	art	
Object Movement	Mark		Item management (containers)	Add/drop item from container	Bug Fixes				Bug Fixes
Communicator Window	Runa			Basic	Bug Fixes	update	Look display	art	Bug Fixes
Slime object	Corinna	Basic	Update	collision /player stuck	Bug Fixes	suppressant			Bug Fixes
AI - Slime	Corinna		Design	replicate		Anima-tion			Bug Fixes
AI - Bandits	Mark		Design			Search/ get objects	Attack	Mission	Bug Fixes
AI - Crew Members	Runa		Design	Logic for travel		Add to game			Bug Fixes
Game Branding, splash screens	Mark	Identify Game Name, Colors, font	Develop content for Login/Home Pages			Find/ Develop art	Find/ Develop art	Add to game	
Character Art	Runa				Find/ Develop art	Find/ Develop art	Find/ Develop art	Final	
Non-Character Art Items	Runa Corinna				Find/ Develop art	Find/ Develop art	Find/ Develop art	Final	
Game State	Runa			Define Basic Game Conds	Update Game End Conds	Update Game End Conds/Handle Randomizing Items in Containers	Handle Difficulty Modes/Handle Randomizing Items Needed for Escape Pods		
Unit Testing	All	Determine environment and methods							

Integration Testing	All	Ongoing	Ongoing	Ongoing	Ongoing	Ongoing	Ongoing	Ongoing	Ongoing
User Trials	All		Establish Trial Procedures		All test for mid-point check	Client Feedback		Friends and family test 1	Friends and family test 2

*Basic: preload, create, update placeholder, render (show object in game)

Individual Development Schedules

Mark		
Week	Planned Activities	Estimate
3	Setup Development Environment Setup Production Environment including required applications Establish GitHub Repository with collaborator access Establish user access for production environment Explore Phaser functionality Login and authenticate user Landing page, create new game button Prepare Progress Report Video Create database with User table Create Trello board for story tracking Develop Room construct	20
4	Add game save functionality to database interaction Determine character movement logic globally and locally Define basic object interaction method for moving Establish production environment snapshot backup procedure Determine game name Design game logo Determine game branding colors Develop method for loading room data from file Prepare Progress Report Video	20
5	Conduct testing for game state save and retrieval from file Design and develop admin window graphics Determine room catalog and storage options Determine item catalog storage options Prepare Progress Report Video	15
6	Create AI construct for bandits	20

	Refine administration menu with transitions Determine bandit slime details Test initial bandit AI attack function Finalize room construct with game object Test and release production version Production deployment of prototype	
7	Test bandit AI with crew member collision Create infrastructure for leaderboard tracking Prepare Progress Report Video Availability Pending - Develop multiplayer framework	10
8	Implement game end state storage with user statistics Build leaderboard report structure Application testing and refinement Conduct user testing Improve interaction with menu Refine menu art Availability Pending - Implement multiplayer framework with data storage Prepare Progress Report Video	15
9	Application testing and refinement Availability Pending - Test multiplayer framework with users Prepare Progress Report Video	10
10	Production deployment of final version Conduct user testing Bug fixes and refinement Report Document	10
Total		120

Corinna		
Week	Planned Activities	Estimate
3	Setup Development Environment Explore Phaser functionality Create basic game.js file with game loop. This file will be used to run the game. Add room(s) maps to the game as tilemap. Ideally generic so changes to map can be made later. Create player object with preload, create, update placeholder, render functions (incomplete) Create slime object with preload, create, update placeholder, render functions (incomplete) << based on work done in week 2	15

	Prepare Progress Report Video	
4	Add Save button to the game file. Save should POST to server. Add placing container and player objects on the map Add collision for walls in rooms (player/slime object collides) Create template for item objects preload, create, update placeholder, render functions (incomplete) Improve update functions for player (movement) Improve update functions for slime (replicate) Design slime AI, get team buy in on how it should behave Prepare Progress Report Video	15
5	Extract game elements to start game from a “new game” file with random object placement Update room object collision to include containers in room Add player motion in response to user selected destination Update slime object collision with player model, add player stuck state Create basic item set from template Implement slime AI Prepare Progress Report Video	15
6	Integration testing Bug fixes for prototype release Create additional items (not for prototype) Load game from previous (not for prototype) Instructions for prototype	15
7	Add multiple team members Add notification of communicator with game state info Create additional items Add suppressant behaviour to slime object Prepare Progress Report Video	15
8	Improve art Map zoom out Actions/Notifications from game to communicator (sounds?) Add player knocked out behavior Prepare Progress Report Video	15
9	Improve art Get user (friends and family) feedback Prepare Progress Report Video	15
10	Final art Instructions for game use Get user (friends and family) feedback Final bug fixes Report Document	15

Total		120

Runa		
Week	Planned Activities	Estimate
3	Setup Development Environment Explore Phaser functionality Design a generic room as a basic tile map and define rooms Determine scale of map Initialize attributes and determine placement of containers Prepare Progress Report Video	15
4	Prepare Progress Report Video Place collidable objects/containers on map Handle dimensions of the containers on the map Design crew members	15
5	Prepare Progress Report Video Handle logic for travel for crew members Define functions to update the containers and collision Design and define basic communicator interface Design and define basic game end conditions	15
6	Instructions for prototype Bug fixes for prototype release Integration testing Find and/or develop art for character and non-character art Update game end conditions	15
7	Prepare Progress Report Video Add crew members to the Game Handle the update of information on the communicator Handle randomizing items in containers Update game end conditions	15
8	Prepare Progress Report Video Define the look and display functions on the communicator Handle randomizing items needed for escape pods Design and define difficulty modes	15
9	Prepare Progress Report Video Get user (friends and family) feedback Make adjustments based on user feedback Handle the display of icons on the communicator and refine design	15

10	Instructions for game use Report Document Final bug fixes	15
Total		120

Conclusion

The Lyra Project team plans to complete an RTS style game that provides the user a unique and entertaining gaming experience. Each member of the team has allocated more than 100 hours towards making this project a success. We have identified a time when team members are available for discussions despite our disparate time zones. Choosing the online collaborative Cloud9 development environment will allow the team to continuously develop and test our project. We have selected the Phaser.io library to provide basic gaming functionality. This selection allows for focus on unique aspects of the game such as game strategies and AI. The result of the development plan should be an on-time and completed project.