Group Eisa Project Plan

Bryan Beabout
James Grunewald
Vijay Kumar

Team Eisa is designing a real time strategy game staged in a post apocalyptic landscape. The game will operate in the browser using HTML5 and JavaScript. Players will be given a choice of factions to choose from and will play against an AI opponent. Players will be able to save the their game, and will be given the option to play a hard and easy difficulty AI. Below we have outlined the broad responsibilities of each team member and their general plan for approaching their portion of the project.

Program Description:

You have found yourself in a post apocalyptic wasteland. Factions fight over mineral resources needed to enhance their armies and expand their territories. You find yourself as the leader of your faction. You must direct the gathering of resources, decide on the placement of new buildings and secure new territories through tactical efforts. The wastelands are laden with roving bands of gun wielding maniacs. In this real time strategy game you will direct units to gather resources, build assets, and attack your opponents.

Program Structure:

The program will consist of an html page with a canvas element that will be manipulated primarily through the use of server side scripts written in JavaScript. Those server side scripts will essential be the game engine that decides how user input is handled and how that is reflected on the canvas element as it loops continuously. In addition the program will have scripts for the AI functionality that will be called from within the main game loop.

We will use a node.js server to handle the initial requests and to load assets from the server as needed. Ideally once a player has loaded the html page all assets will be loaded using “spritesheets” in an efficient manner. Those sprites will be parsed into their independant items and the game engine will display them as necessary. This will help to minimize the number of server calls that need to be made during actual gameplay to reduce latency. Otherwise the only other time the server should need to be contacted is when the player saves or loads a game.

Program Resources:

JavaScript
Node.js
Express.js
HTML5
Tiled
ShoeBox
TexturePacker

Team Member Assignments:

- Bryan Beabout
    - Graphics and Game Engine
        - Graphics
            - Implement the game map
                - Load pre-designed maps
            - Implement game sprites
                - Load pre-designed sprites for buildings, units, and resources
                - Implement animations for units/buildings/resources that have them
            - Implement status indicators
                - Implement indicators that pop-up showing a unit/building's name and health status when selected.
                - Implement indicator that pops up showing a resource's name and resources-remaining status when selected
            - Implement side-menu that allows user to make selections, such as constructing buildings or units and see information about the user's army (resources collected, units-created, etc.)
        - Game Engine
            - Visual:
                - Implement ability to move the viewport around the game map
                - Implement the ability to select units/buildings/resources via mouse and see that entity's current status
            - Interaction from user to game:
                - Implement the ability to make construction decisions (for units and buildings) from the menu limited by how many resources the user currently has available
                - Implement the ability to select an area to construct a building

            - Implement the ability to give orders to units (movement, attack, harvest)
            - Allow user to store/retrieve game state
        - Interaction between objects within the game:
            - Projectiles
                - Implement ability for some units to fire projectiles
            - Collision detection
                - Implement inability to move past some resources
                - Implement damage from projectiles
        - Interaction with other teammate's systems:
            - Read and update information about units/buildings/resources from influence maps and unit/building/resources data files
            - Read sprite information from unit/building/resources data files
            - Utilize AI libraries to implement computer-opponent decisions
            - Utilize AI libraries to implement user unit movements across map
- James Grunewald
    - Server and Database
    - AI System
        - Influence Maps
            - Builds matrices that represent various game states
                - Player's Mobile Units
                - AI's
                - /Mobile Units
                - Player's Buildings
                - AI's Buildings
                - Resources
                - Buildings/Barriers
        - Resource Management
            - Decides how to manage units that can gather resources
            - Uses data from the resource influence map and the Asset Management Module and

Military Module to determine which resources to gather
        - Asset Management Module
            - Decides which buildings or defensive structures to build next by managing a queue
            - Manages builders
            - Repairs buildings
        - Military Module
            - Determines ideal targets based on clustering of enemy units from the building and mobile unit influence maps
            - Determines the best units to build based on the ideal targets
            - Grouping Module
                - Groups units into clusters to minimize weaknesses
            - Decides which units are going to attack and where
                - Does this by determining minimum DPS required to destroy ideal targets
                - Also decides if attack should be postponed based on movement of enemy units
        - General Path Finding Algorithm
            - Decides how a unit will move from point A to point B
            - Attempts to avoid routes that will expose units to enemies
                - Exception is if the purpose is to attack
            - Minimizes distance as a last priority
            - Avoids known obstacles
        -
    - Project Plan
- Vijay Kumar
    - Mid Report and Final Report
    - Units and Balance Developer
        - Design all Entities in game
            - Create theme for all objects in the game world. This includes and is not limited to:

    - Buildings
        - Factory
        - Home-Base
    - Structures (Walls/Towers)
        - Base Tower
        - Missile Tower
        - Wall
    - Players
        - Engineers
        - Fighters
        - Mutants
    - Vehicle
        - Car
        - War Bus
        - Semi-Truck
        - Super Mutant (comparable to veh)
        - Giants
        - Monster

- Create objects and their respective functions for proper behavior for all units, players and other structures
- Create update functions for each entity
    - Update function will allow Game Engine to properly draw all objects to the canvas at appropriate times
- Create kill function for each entity
    - The kill function will remove an object when it is ready for removal
- Develop all units to interact in a balanced manner
- Create collision functions to properly manipulate object data when a collision is detected.
- Create class hierarchy

Beginning Outline of Game Objects/Units

Clan 1: (The Academic Class)

Resources:

“Futuristic” Fuel: (to be renamed later)
- Each engineer can collect 20 gallons of fuel at a time

The Lab: (available at onset of game)
- can train engineers
- can train one engineer per minute
- costs 500 mineral points for each engineer

Engineers: (5 available at onset of game)
- Gathers minerals and builds structures
- Each engineer can hold 20 gallons of fuel at a time
- Must collect from deposits and return to The Lab

Basic Training Center:
- A single engineer can build
- Can be built in 30 seconds by an engineer
- costs 500 to build basic training center
- Initially creates fighters with 10 points of health
- creates 1 fighter every 30 seconds

Upgrade Training Center to:

Intermediate Training Center
- A single engineer can update
- Can be updated in 30 seconds
- costs 1000 to update from Basic Training Center
- creates fighters with 30 points of health
- creates fighters every 60 seconds

Advanced Training Center
- A single engineer can update
- Can be updated in 30 seconds
- costs 2000 to update from Int. Training Center
- creates fighters with 60 points of health
- creates fighters every 60 seconds

Factory:
- Can create armed vehicles
- A single engineer can build
- Can be built in 30 seconds by an engineer
- costs 200 mineral points

- Factory first builds armored car which:
    - Armor value of 10
    - inflicts 10 pts of damage with gun power
    - cost is 300 mineral points
    - builds 2 cars per minute

Upgrade Factory to:

War Bus -
- Armor value of 20
- inflicts 20 points of damage with gun power
- Factory Upgrade cost 600 mineral points
- Builds 1 bus per minute

Semi Truck -
- Armor value of 40
- inflicts 40 pts of damage with gun power
- Factory Upgrade cost is 1000 mineral points
- builds 1 Semi Truck every minute

Towers:
- Can be built by 3 engineers
- Cost 2000 gallons of fuel to construct
- Have an armor rating of 300
- Can inflict 20 pts of damage via gun power

Upgrade to:

Missile Tower:
- Can be upgraded by 3 engineers
- Cost 2000 gallons of fuel to upgrade from Baseline Tower
- Armor rating of 400
- Can inflict 40 pts of damage via gun power

Rebuild Wall:
- Can be rebuilt by 3 engineers
- Cost 1000 gallons of fuel to rebuild tower

Walls:
- Can be built by 3 engineers
- Cost 2000 gallons of fuel to construct
- Have an armor rating of 100

Rebuild Wall:

- Can be rebuilt by 3 engineers
- Cost 1000 gallons of fuel to rebuild wall

Clan 2: (Working Class Clan)

Resources:

Nuclear Waste: (to be renamed later)

- Each can mutant-human collect 20 gallons of fuel at a time

Nuclear Plant: (available at onset of game)

- can create mutant-human
- can create one mutant-human per minute
- costs 400 gallons of fuel for each mutant-human

Mutant-Humans: (10 available at onset of game)

- Gathers mutant-human and builds structures
- Each mutant-human can hold 20 gallons of fuel at a time
- Must collect from deposits and return to Nuclear Plant

Basic Training Center:

- A single mutant-human can build
- Can be built in 30 seconds by an mutant-human
- costs 500 to build basic training center
- Initially creates fighters with 5 points of health
- creates 1 fighter every 30 seconds

Upgrade Training Center to:

Intermediate Training Center

- A single mutant-human can update
- Can be updated in 30 seconds
- costs 1000 to update from BasicTraining Center
- creates fighters with 15 points of health
- creates fighters every 60 seconds

Advanced Training Center

- A single engineer can update
- Can be updated in 30 seconds
- costs 2000 to update from Int. Training Center

- creates fighters with 30 points of health
- creates fighters every 60 seconds

Factory:

- Can create super mutants
- A single mutant-human can build
- Can be built in 30 seconds by an mutant-human
- costs 200 mineral points
- Factory first builds armored car which:
    - Armor value of 8
    - inflicts 8 pts of damage with gun power
    - cost is 300 mineral points
    - builds 2 mutants per minute

Upgrade Factory to:

Mutant Giants -

- Armor value of 15
- inflicts 15 points of damage
- Factory Upgrade cost 600 mineral points
- Builds 1 Giant per minute

Monster -

- Armor value of 32
- inflicts 25 pts of damage with gun power
- builds 1 Monster every minute

Towers:

- Can be built by 3 engineers
- Cost 2000 gallons of fuel to construct
- Have an armor rating of 300
- Can inflict 20 pts of damage via gun power

Upgrade to:

Missile Tower:

- Can be upgraded by 3 engineers
- Cost 2000 gallons of fuel to upgrade from Baseline Tower
- Armor rating of 400
- Can inflict 40 pts of damage via gun power

Rebuild Wall:

- Can be rebuilt by 3 engineers
- Cost 1000 gallons of fuel to rebuild tower

Walls:

- Can be built by 3 engineers
- Cost 2000 gallons of fuel to construct
- Have an armor rating of 100

Rebuild Wall:

- Can be rebuilt by 3 engineers
- Cost 1000 gallons of fuel to rebuild wall

James Grunewald's Timeline (estimated 10+ hours each week):
Week 3: Research and Refinement of algorithm choices/Incorporate influence maps
Week 4: Build the state machine for AI unit behavior/Develop Path Finding function
Week 5: Develop Resource management Module
Week 6: Develop Asset Management Module
Week 7: Develop military module
Week 8: Build Grouping Module
Week 9: Testing and training of AI
Week 10: Enable Multiple Difficulty Options

Vijay Kumar's Timeline:(estimated 10+ hours each week):
Week 3: Final Design of all objects used in gaming environment (ie units, players, etc)
Week 4: Rough draft of object classes and overloaded update functions, etc.
Week 5: Implementation of collision function for each object
Week 6: Implementation of kill function for each object
Week 7: Game Engine integration of all objects
Week 8: Finalize all object classes
Week 9: Ensure proper function of all objects within the game environment
Week 10: Test and ready for submission

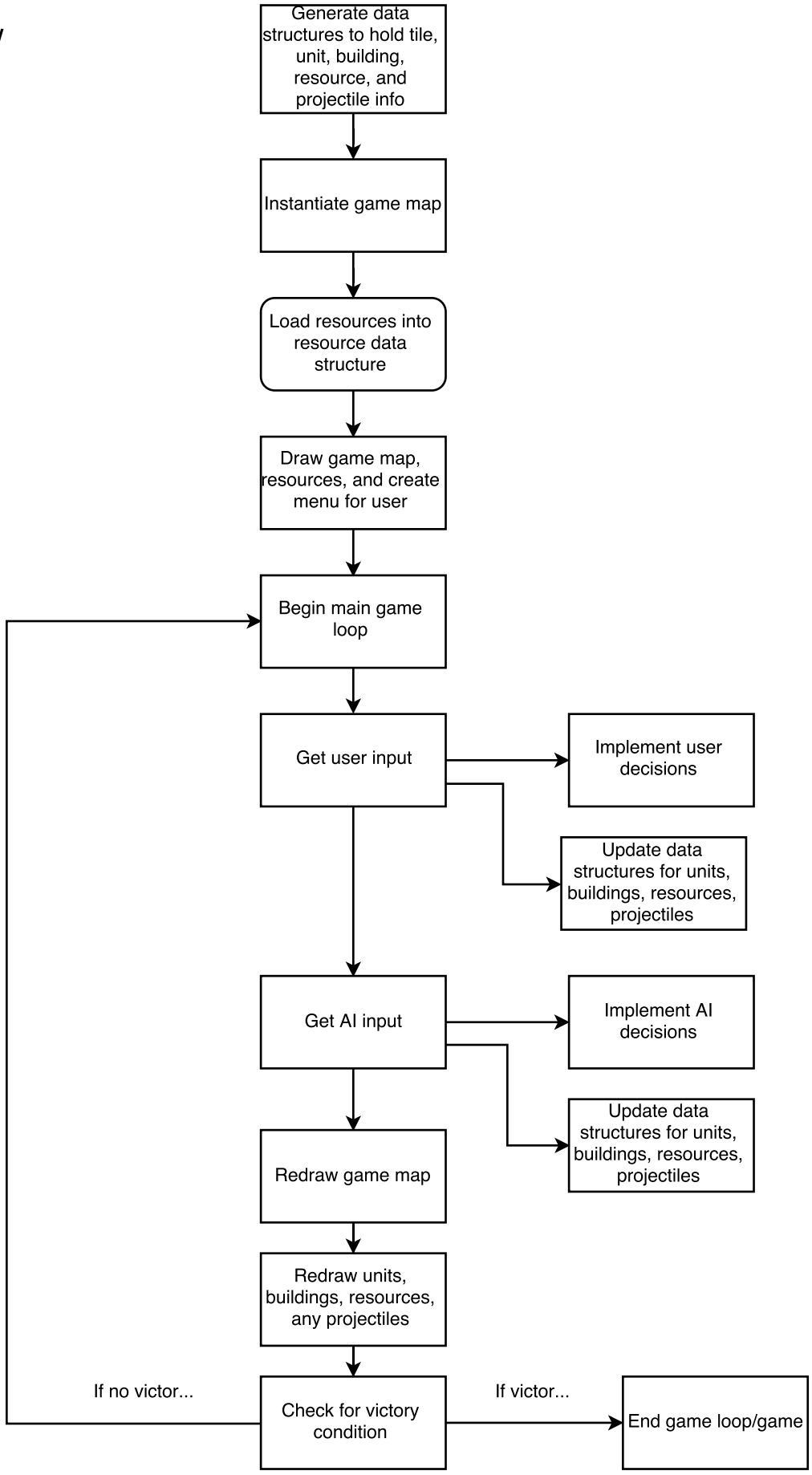
Bryan Beabout's Timeline (estimated 10+ hours each week):
Week 3: Develop canvas, add game viewport, allow movement around map
Week 4: Develop loading the game map, game objects, add game states (will be modified to integrate with James and Vijay's work)

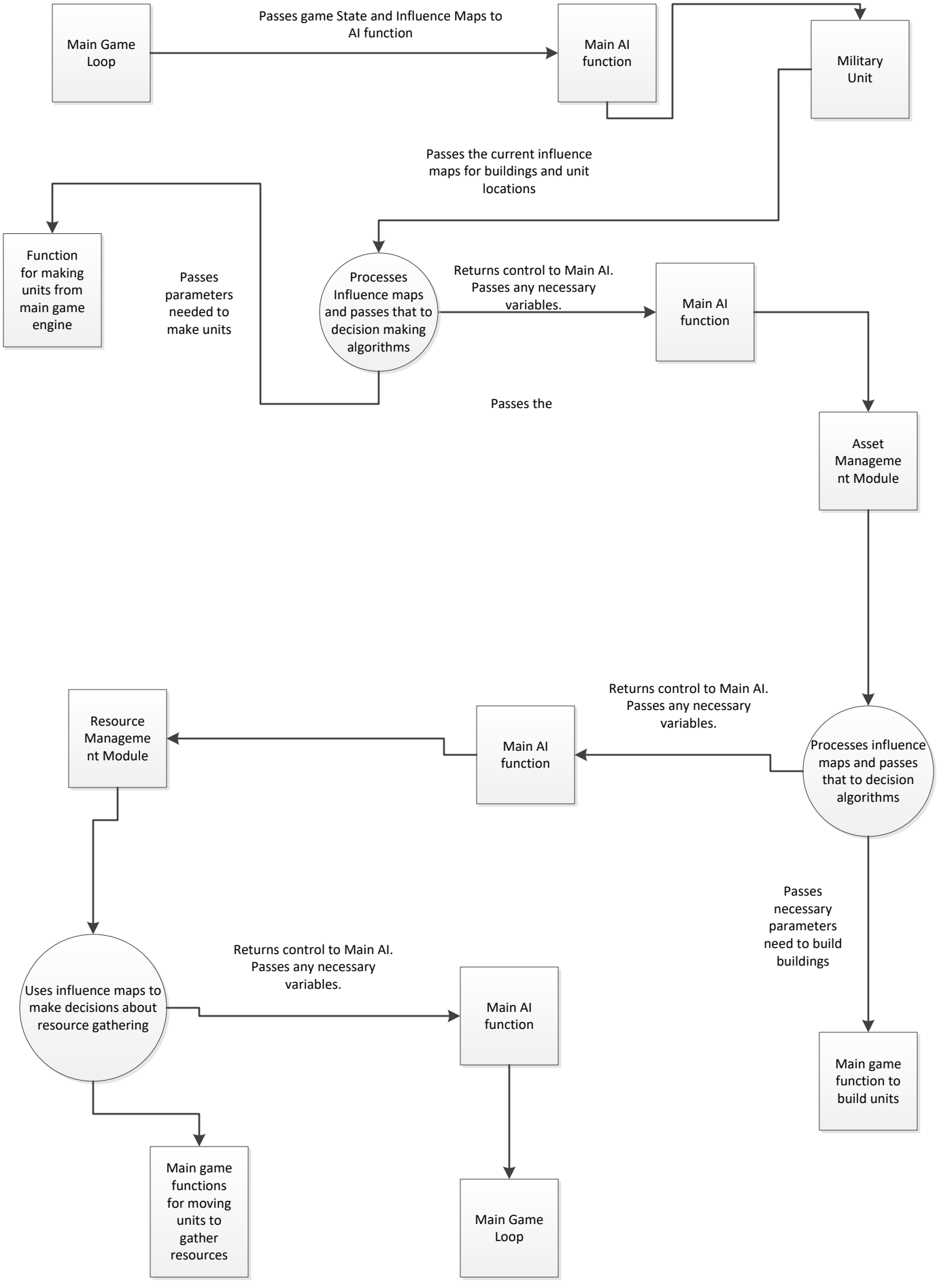
Week 5: Develop contextual unit/building construction menu
Week 6: Develop user interaction with game map, game units (will be modified to integrate with James and Vijay's work)
Week 7: Implement status pop-ups for unit/resource/building selection
Week 8: Continue integration with Vijay's work, utilize Vijay's modules to load game objects, provide interactions between units
Week 9: Continue integration with James' work, integrate AI modules with object statuses
Week 10: Test, test, test

Throughout development of the game, each team member will work closely with the other developers to ensure consistency in our respective engines. In addition, we will continually develop the protocol that regulates how the Game Engine communicates with the AI and game objects. By working through this term with strong communication we feel confident that team EISA will create a smooth gaming experience.

In the event any one developer has the availability to co-pilot a portion of another developer's assigned tasks, we will take the necessary steps to ensure an evenly distributed workload.

Game Engine Flow




Main Game Loop
Passes game State and Influence Maps to AI function
Main AI function
Military Unit
Passes the current influence maps for buildings and unit locations
Function for making units from main game engine
Passes parameters needed to make units
Processes Influence maps and passes that to decision making algorithms
Returns control to Main AI. Passes any necessary variables.
Main AI function
Passes the
Asset Management Module
Resource Management Module
Main AI function
Returns control to Main AI. Passes any necessary variables.
Processes influence maps and passes that to decision algorithms
Passes necessary parameters need to build buildings
Uses influence maps to make decisions about resource gathering
Returns control to Main AI. Passes any necessary variables.
Main AI function
Main game function to build units
Main game functions for moving units to gather resources
Main Game Loop