Introduction

Austin Cheng, James Davis, Jeremy Ethridge, Andrew Fung, Huy Le, Juan David Lopez, Helen Pabst, Alex Tol

Project Scope

Seize Command is a top down multiplayer space game which encourages players to use communication and teamwork in order to upgrade their ship and eventually overcome the enemy. In order to improve, players will need to risk leaving the safety of their ship and board the enemy to either take control or salvage parts to improve their own vessel.

The intent of this project is to introduce a unique mix of exciting gameplay elements never before seen in the space sim industry while providing a nostalgic feel for veteran players of the genre.

A variety of features which will be implemented include but are not limited to:

- Multi-Player Support
- Animations and Sound Effects
- Particle Systems and Special Effects
- Intuitive User Interface

Seize Command aims to present the video game community with an intensely entertaining addition to the space sim genre.

Major Software Functions

- **Multi-Player Support**
    - Our product will target those looking for a cooperative and competitive experience and will therefore require Multi-Player networking support. We anticipate that a large amount of troubleshooting with regard to server and network configurations will need to take place considering our teams lack of experience in the field. Unity provides easy setup for Multi-Player features which should speed up research related to Networking.
- **Interactive Desktop Application Provided by Unity Engine**
    - Unity 2D provides easy to use build settings allowing for a Desktop application to be created with the click of a button. There could exist optimization issues with regard to different versions of the windows operating system. Ideally we will have to iron out these issues for specific versions and decide which others will be deemed incompatible or “too old.”
- **Enemy Ship Boarding**
    - One of the main selling points of this product is the ability for a team to board an enemy ship. The point of this action is to provide a “high risk high reward” situation that a player's team can undertake. This function offers players to make

a vital choice while playing: Do they destroy the enemy ship and move on or risk salvaging for better equipment.

- **Clear and Straight-Forward Tutorial**
    - New players will have the option to experience a tutorial mode which will bring them through the main functions of the game.
- **Non Player Characters**
    - Non Player Characters(NPC) will be included as enemies for the players to overcome. Our team will be developing a baseline Artificial Intelligence(AI) for the NPC to function off of. The complexity of the AI will likely vary from enemy to enemy ranging from very basic fighters to full sized enemy frigates.
- **Sound Effects**
    - Spaceships have a variety of functions and necessities that will require accurate sound effects. For instance, functions such as a ship firing its weapons, activation of thrusters and player interactions would be required to have sound effects. It is also important to consider that no sound effects should sound too similar to any other. This would likely vary from ship to ship.
- **Animations**
    - Much like sound effects, animations will be needed to accompany most player driven functions within the game. Uniqueness of each animation is key so that one function is not mistaken for another. The visual accuracy of each animation will be a challenge and research into Unity’s Particle Effects will need to take place.
- **Game Modes**
    - **Team Death Match**
        - Team Death Match is a team vs team game mode where strategy and team player mechanics will determine victory over the other team.
    - **Cooperative**
        - In Cooperative, one team takes on the enemy fleet of AI. The player team starts off with a very low end ship. The goal is to continue to get stronger via better parts and ships. Should a team acquire enough strength. They can take on the AI centralized space station in order to win the game.
- **User Interface**
    - The player’s User Interface(UI) will be intuitive and allow for seamless interaction with the player’s ship and controls. The UI should also be uncrowded so as to enhance the user's visual experience as much as possible
- **Leaderboards**
    - Players will be able to compare their scores with other players via the player leaderboards which would display personal statistics. Some of the statistics would be but not limited to: username, score, date.

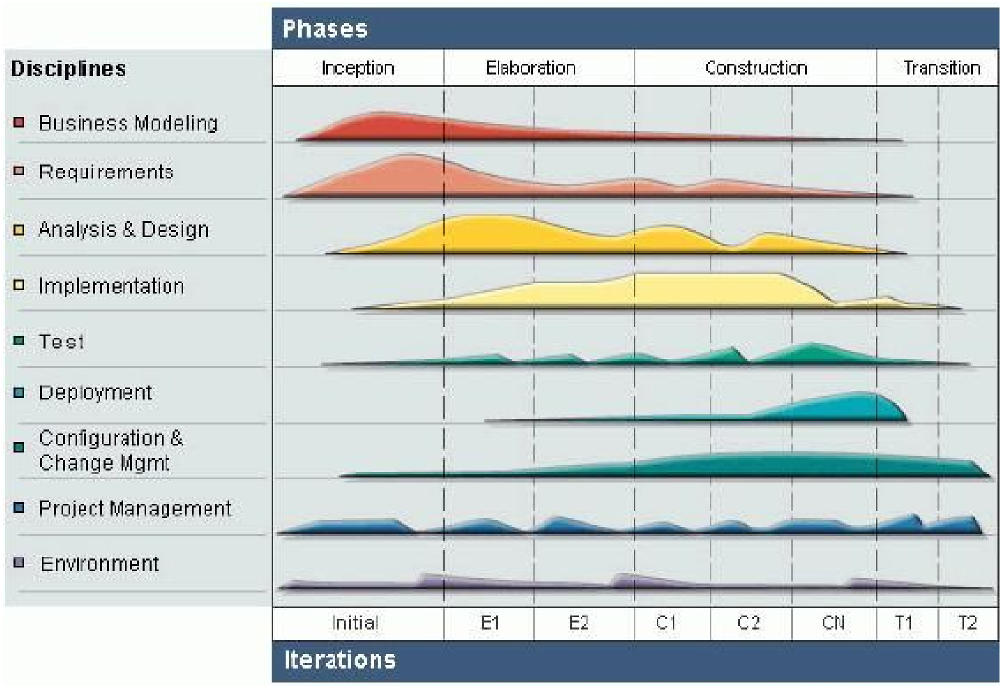
- **Database**
    - A database will be required in order to store leaderboard statistics as well as in game items values. Some of these values might include but are not limited to: weapon damage, ship health, engine speed.
- **Webpage**
    - Our team will make a website to allow for user registration as well advertising for the product. In addition, player leaderboards will be displayed on the website for players to view. Finally, the game will be available to download on our company webpage.

Performance/Behavioral issues

The game itself is a top down 2D space sim, so all modern computers and most aged computers (earliest from 2004) should be able to handle the game. The most graphically taxing aspect of the game will be the numerous bullets/lasers/torpedoes that will be on screen, but proper coding and animation work should make this negligible. The game will have a multiplayer aspect which can be perfectly executed as long as optimization is handled properly. The only thing that will be out of the control of the game will be the user's internet connection, if the user's connection is shoddy then the user will likely experience trouble connecting to the game.

Management and Technical Constraints

Although this team will endeavor to finish every task on time or even early, there will most likely be unforeseen problems and new requirements in the future. As a result this team will likely be working on multiple tasks during certain weeks. Therefore, we will be using the Unified Process model as this model is most suited to our situation. For this project there should be very little technical constraints, we are making a 2D game so rendering and running the game should be a non-issue. The biggest technical hurdle for this game will be the multiplayer aspect, however since the game itself is not resource intensive the only real challenge will be learning to use the network manager.



Project Estimates

Project Resources

Our team consists of eight members. Each member will take charge of a certain aspect(s) of the game development. This by no means states that a member will only work on their parts based on their role. They will assist each other; however, any questions or concerns regarding a part of the project, the member in charge will manage the situation.

Required Staff

- Graphic Designer
- NPC Developer
- Lead Unity Programmer
- Assistant Unity Programmer
- UI Designer
- QA
- Website Developer
- Animator
- Audio Engineer
- Game Programmer
- Documentarian

Required Hardware

- Minimum Requirements
    - Intel Processor

        - Integrated Graphics
        - 8 GB RAM
        - 250 GB HD

Required Software

- Unity
- Windows 10
- Microsoft Visual Studio
- Filezilla

Risk Management

Project Risks

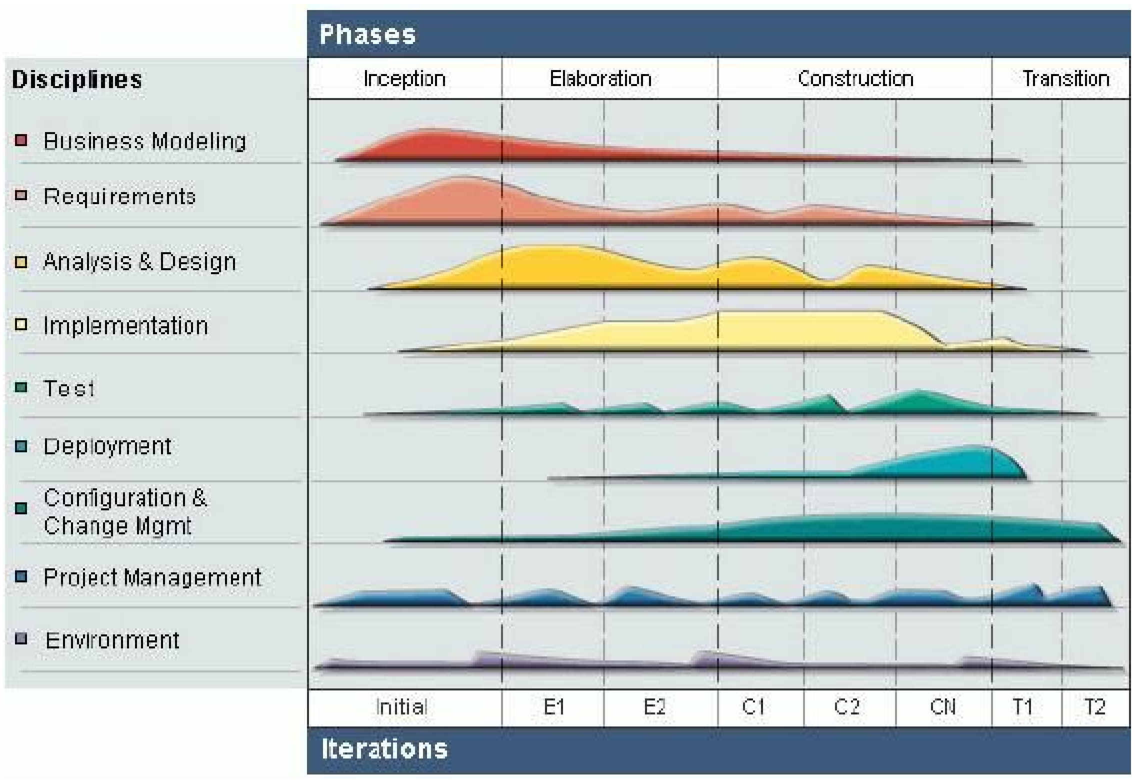
Major risks that may hinder the completion of this software project:

- Insufficient coding experience for project development and potential issues that arise
- Inadequate knowledge/experience of Unity or C#
- Poor project management and task scheduling
    - Bad time management could lead to missed deadlines
- Poor group communication and work distribution
    - Potentially caused by too many contributors
- Too many features planned or features that are too difficult/ambitious to properly implement

Project Schedule

Project Task Set

As aforementioned, we will be using the Unified Process Model.



Framework Activities

- Planning/Design
- Risk Analysis
- Programming
- Testing
- Professor's Evaluations

Task Set

- Requirement Specification
- Learning Unity
- User Interface Construction
- Artwork Design and Editing
- Backend Game Rules Construction
- Database Construction
- Game/Software Manual Construction
- Website Construction
- User Login Construction
- Testing

List of deliverables

Documentation

- Vision Document
- Project Plan (this document)
- Flowchart
- UML
- Use Cases
- Test Cases
- User Manual

Code

- Game Interface Mockups
- Game Interface Database
- Game Playable Prototype (Alpha)
- Game Complete Interface
- Game Prototype with Assets (Beta)
- Game Final Build
- Website Interface Mockup
- Website Complete Interface
- Complete Product

Functional Decomposition

Game Rules Task Breakdown

- Universal
  - Objects are destroyed upon health points reaching zero
  - Players only control one command console at a time
- Coop
  - Boarded ships switch command to player control upon player(s) holding bridge for pre-determined amount of time
  - Medical bay ends respawning of NPCs upon loss of ship control
  - Parts can be removed from ships
  - Parts can be implanted into player's ship(s)
  - NPCs (ships and crewmembers) attack player(s) based on predetermined attack pattern and targeting algorithm
  - NPC ships patrol within predetermined zones
  - Player(s) lose game upon player ship destruction
  - Player(s) win game upon capturing center space station
- Team Deathmatch
  - Pre-game team declaration
  - 1 point is awarded to team that destroys an enemy ship
  - 3 points are awarded to team that boards and plants a bomb on enemy ship
  - Team that gains 10 points wins
  - Team loses when opposing team gains 10 points

Game User Interface Breakdown

- Ship animations/sound effects
- Player animations/sound effects
- Weapon animations/sound effects
- Background art and ambience
- Mouse cursor sprite change based on which console is being used
- Menu to exit or change options

Website User Interface Breakdown

- Fully functional user login
- Access to the leaderboard
- Ability to download the Seize Command

Testing Breakdown

- Game needs to be played thoroughly to check for bugs
- Network needs to be tested thoroughly to test stability
  - Will need to test maximum network load
- Website user login and database must be tested for memory leaks and low-level security breaches

	Sep 20	**Sep 27**	**Oct 4**	**Oct 11**	**Oct 18**	**Oct 25**	**Nov 1**	**Nov 8**
Documents								
Flow Chart	🟦	🟦						
UML		🟦	🟦					
Use Cases			🟦	🟦	🟦			
Test Plan					🟦	🟦	🟦	
User Manual							🟦	🟦

Sandbox	**Nov 15**	**Nov 22**	**Nov 29**	**Dec 6**	**Dec 13**	**Dec 20**
Player Control	🟦	🟦				
Ship Control		🟦	🟦			
Basic Graphics	🟦	🟦	🟦			
Camera Change			🟦	🟦		
Projectiles				🟦	🟦	
Destructible Objects				🟦	🟦	
Basic Boarding				🟦	🟦	🟦

Graphics	**Feb 7**	**Feb 21**	**Mar 7**	**Mar 21**	**Apr 4**	**Apr 18**	**May 2**
Sprites							
Ships	🟦	🟦					
Weapons			🟦				
Projectiles			🟦				
Space background			🟦	🟦	🟦		
Space Station				🟦	🟦		
Player					🟦		
Animations					🟦	🟦	
Sound Effects						🟦	🟦

Code	**Feb 7**	**Feb 21**	**Mar 7**	**Mar 21**	**Apr 4**	**Apr 18**	**May 2**
Team Deathmatch							
Ship Functionality							
Roles	🟦						
Camera Functionality	🟦	🟦					
Player Combat		🟦					
Shield/Hull Points		🟦	🟦				
Win Condition							
Ship Destruction Point			🟦	🟦			
Bomb Plant Point			🟦	🟦			
Respawn				🟦			
Ship Database				🟦			
Coop							
Boarding Functionality				🟦			
Parts				🟦	🟦		
Parts Database				🟦	🟦		
Win Condition					🟦		
Map					🟦	🟦	

Generation					🟦	🟦	
Networking				🟦	🟦	🟦	🟦
AI				🟦	🟦	🟦	🟦
Testing						🟦	🟦
Website						🟦	🟦

Staff Organization

The following describes the specialized roles and tasks each team member will partake in for this software engineering project. These roles do not constrain the tasks that can be done by each team member; members will work amongst each other and will provide support outside of their mentioned area of expertise.

Austin Cheng: Austin Cheng will be the game's Test Engineer. Cheng will test each of the software components and search for any possible ways to break those components. Any issues that are discovered will be reported to the project's GitHub Issue tracker.

James Davis: James Davis will be *Seize Command's* secondary Unity programmer. Davis, already having had experience with Unity, will take charge of this role.

Jeremy Ethridge: Jeremy Ethridge is the project manager. Ethridge is responsible for coordinating combined engineering efforts amongst team members, who each have their own special niche. Ethridge coordinates the communication between the team members.

Andrew Fung: Andrew Fung's expertise in artificial intelligence and algorithms grants him the role of programming artificial intelligence algorithms in the game's NPCs (non-playable characters). Fung's role is quite crucial as a major factor of *Seize Command's* difficulty and playing experience derives from the NPCs.

Huy Le: Huy Le is *Seize Control's* Graphic Artist. Huy will be responsible for acquiring and designing necessary game art for game entities and features such as the background, enemy ships, player ships, and other NPCs.

Juan David Lopez: Juan David Lopez is *Seize Command's* User Interface Engineer and Web Developer. Lopez will be responsible, in coordination with Le, to graphically design game components and layout what will be visible to a user during gameplay, such as any radars, space viewers, and any internal spaceship controls. Lopez will also be the lead web developer for the game to have an online web presence, serving as a form of media for *Seize Command* to be advertised and downloaded.

Helen Pabst: Helen Pabst will be *Seize Command's* main Unity programmer. Helen's experience with the Video Game Development Association has given her the technical skills necessary to take this role.

Alex Tol: Alex Tol is *Seize Command's* Networking and Database Engineer. Tol's current growing knowledge in networking and database design concepts serves him this role quite well. Networking allows the game to have a multiplayer feature, and a database grants players to save their high scores. Tol will design the database using UML concepts and implement a backend with NodeJS.

Tracking and Control Mechanisms

Quality Assurance and Control

- Before the implementation of the game, the team must have analyzed and tested all the use cases available to ensure the workflow of the program.
- Another testing period begins when we have a prototype of the game. Our team of 8 people will apply all the use cases and testing out different aspect of the game like co-op and online multiplayer to ensure the network quality of the program.
- Any bugs that are found will be reported and fixed before the actual release of the game. After that it will become an on-going task of keeping track on the quality of the program through users' experiment and release updates if necessary.

Change Management and Control

- All the documents relate to the project will be updated on GitHub. The team will use GitHub as a project management to keep record of all the changes.
- GitHub allows us to post code and manage on what part we are currently working on and have been done of all team members.