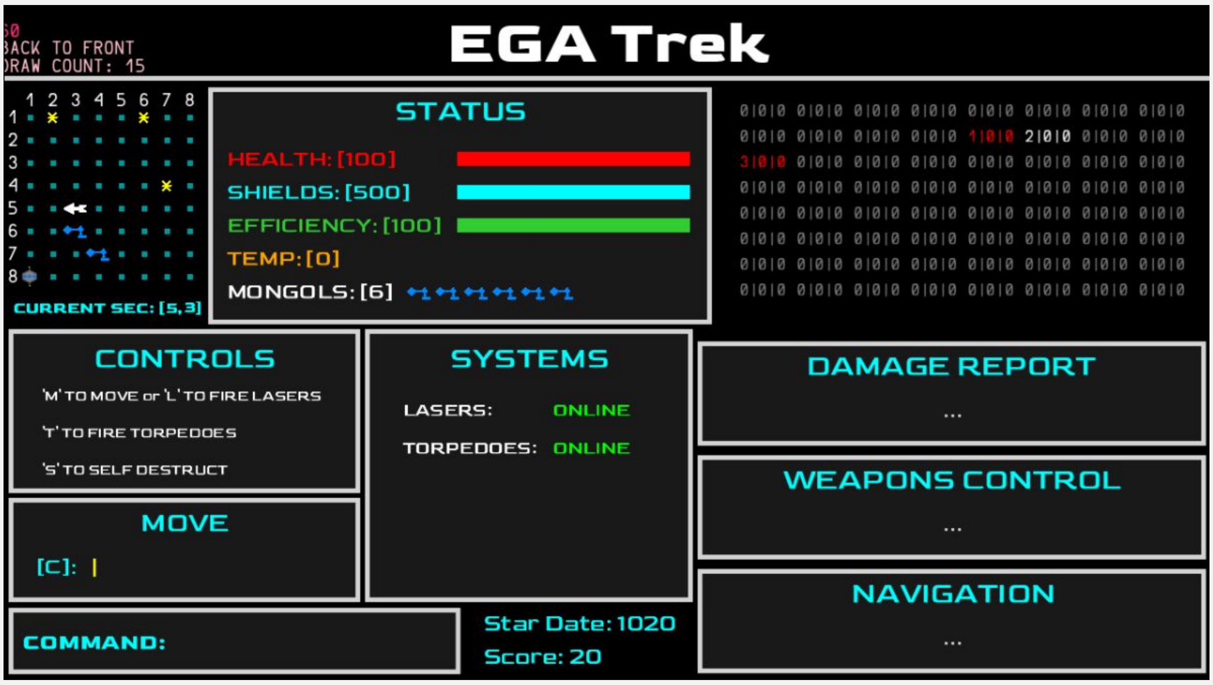
More Games In C++

Charlie Joe Evans 08009251

EGA TREK

Game Design and Technical Design Document (GDD/TDD)



Contents

GAME DESIGN DOCUMENT

OVERVIEW

ELEVATOR PITCH / BRIEF DESCRIPTION

My game will be a modern adaptation and port of EGA Trek, an updated and graphical upgrade to the classic videogame of the 1980s game Star Trek which was a text based, bridge commander-style strategy game. Just like EGA Trek, the game will task the player with hunting down and destroying all mongols located in the sectors and quadrants in order to attain victory. The game will retain a space setting with simple graphics.

CORE GAMEPLAY MECHANICS

MOVEMENT

Sector

By entering 'MOVE' into the command box, the move command box will be brought up. To move to a position in the sector map the player will need to specify a grid coordinate. It will ask the player for the Y axis coordinate first before asking for the X. After entering a coordinate, making sure that a comma separates the two values, the player will be transported to that location in the sector grid.

Quadrant

To move in the quadrant grid, I will be implementing a new system to that which is present in EGA trek. Instead of asking the player to input a further set of coordinates after the sector grid coordinates i will separate the movement entirely from the sector grid movement. Utilising the arrow keys the player will be able to traverse through each of the quadrants and hover over the quadrant they wish to be situated in. This will be the player's quadrant position. If the quadrant contains mongols then that will be highlighted in the sector grid, rendering the amount of mongols that are present in that quadrant.

SHOOTING

Lasers

To fire the lasers the player should enter ‘LASER’ into the command box. However, like with all other commands, the first letter of the command will suffice. Once this is entered the laser command box will be brought up and after specifying a grid coordinate in the sector map (like that of the movement mechanic) lasers will be fired at that space. If a mongol is present then damage will be dealt to, so long as the player is also in its vicinity.

Torpedos

The torpedoes will be fired exactly the same way to the lasers with the only differences being that the player needs to type ‘TORPEDOS’ into the command panel and also the effect on the players gameplay experience like demanding more energy but dealing increased damage.

REPAIR

When the player takes too much damage they will need to repair their ship. To do this, instead of typing in a command, the player will need to move to a location in the sector map that is in the vicinity of a starbase. So long as the player’s health isn’t full already, the player will regain health each turn so long as they keep moving around the base. Staying idle in the same grid location will not continually increase health. The player will need to keep in mind that if a star is also in the vicinity, this will counteract the healing effect and cause nothing to happen. To counter this the player must make sure they are in a safe location in proximity of the base to gain its benefits.

SELF-DESTRUCT

Should the player need to leave the game or give up, they will need to type ‘SELF DESTRUCT’ into the command box to bring up the self destruct command box. Here if the player enters their self destruct password, which they provided back in the menu, the game will bring the player to the defeat screen and the game will end.

INFLUENCES

Victory and Defeat Text

The style and design of the victory and defeat screen text will be inspired by a similar style present in another classic video game, Star Wars Battlefront 2 (2005). Here, upon victory or defeat, big bold coloured text would flash to the screen letting the player know of their subsequent victory or defeat. I will have a short animation that is also influenced by Star wars once again with how at the beginning of all star wars movies at the start of the text crawl the title would flash out of nowhere really large and promptly shrink smaller and smaller until it was no longer visible. I want to do the opposite of this in my game whereby it enlarges out from the center of the screen until it reaches an appropriate size and stops enlarging.

The Overall Design of The Game

The overall style of the game will draw a lot from EGA Trek such as the design of the player, mongols, stars and starbases which remain loyal to the original games sprites. Furthermore, the different panels such as the systems and report panels will be designed similar to that of EGA Trek but with my own fresh coat of paint.

DIFFERENCES TO THE ORIGINAL GAME

MOVEMENT

Instead of specifying a second set of coordinates after the sector coordinates to move quadrants, I will implement a whole new way to move quadrants. This will utilise the use of the arrow keys to change quadrant, highlighting the players current quadrant in white, quadrants with mongols present in red, with the remaining vacant quadrants being a dark grey colour. Hovering over a quadrant with mongols in will render mongols in the sector grid for the player to go and eliminate.

COLOURS

EGA Trek's primary colour was blue with panels outlined in white, white text, and an assortment of different colours was used for the sprites. For mine I will be going for a much darker colour scheme with black being the primary colour, panels will possess a dark grey background with white outline and aqua blue shall be used for the text colour. I have chosen this particular style as I feel it gives the game more of a space-esque feel and tone.

SHOOTING

Instead of specifying damage and how much to deal to all of the mongols at once, my shooting will work by specifying a coordinate and then randomly generating a damage output to be inflicted on the selected mongol only. The torpedoes will generate a significantly larger damage output compared to the lasers. I've chosen to do this because I feel it would bring an improvement to the gameplay if each mongol could be taken out in turn rather than all at once because that is not how an actual battle would play out you wouldn't just take out the entire fleet at once.

STORY / NARRATIVE

BRIEF

"*The Mongols have invaded. They have destroyed our homes and stripped our world of its precious resources. Before long the galaxy will be under Mongol control and the war will be lost. We must fight them back. The resistance summons you. Take out all Mongols clusters that reside in each quadrant and we might just stand a chance against the Mongol invaders and restore our galaxy to its former glory...good luck commander.*"

GAMEPLAY

The gameplay loop will be focused around eliminating the mongol threat. The player will start off the game with full health and shield, with all ship weaponry online. They will need to use the laser and torpedo cannons to fire shots into specific sectors and eliminate the mongols that dwell there. Whether they stray too close to a star or battling a mongol, the player will inevitably lose health and overheat their weaponry and so will need to visit a nearby starbase to heal up. This is how the game will play out until all mongols have been destroyed and the player has either attained victory or suffered defeat.

PROGRESSION AND CHALLENGE

To promote a challenge into the game, I plan to include a series of different difficulty levels. Each level will increase the amount of mongols that spawn into the game, as well as significantly reducing the players starting health and shield. Weaponry will overheat more quickly and stars will become more prevalent across sectors.

CONTROLS

Entering text - Keyboard keys

Left Arrow Key - Move left in quadrant galaxy map (game)

Right Arrow Key - Move right in quadrant galaxy map (game) / flick through brief pages (menu)

Enter Key - To enter commands and close command boxes

Escape Key - Exit game

ASSETS

ART

- Piskel art will be used to create
  - Mongols
  - Stars
  - Starbases
  - Player
  - A simple solid coloured box used for the creation of the games panels and grid points.
- Fonts
  - Goldman-Bold (titles)
  - Goldman-Regular (general text)

CODE

- SCENES
    - SceneManager class
    - Menu class
    - Win class
    - Lose class
- SHIPS
    - Starship class
    - Player class
    - Enemy class
- UI
    - Panel class
    - Grid class
    - Button class
    - EnergyBar
- OTHER
    - Gameobject class
    - Game component class

USER INTERFACE (UI)

HUD

Health

Health starts at 100. When the player's shield has been destroyed, the player will then be able to lose health. Attacking a mongol will take 10 health.Visiting a starbase will restore some health each turn as long as the player moves around in the vicinity of the base. Straying too close to a star 10 health if shields are depleted.

Shields

Shields starts at 500. The shields can be depleted at the start unlike the health. Significantly more shield is lost than health when attacking a mongol or being in proximity to a star. 50 shields are lost for being in a star’s proximity.

Efficiency

Efficiency starts at 100. As long as efficiency is more than 0, the player can make use of weaponry such as lasers and torpedoes. The efficiency also affects how much damage can be inflicted onto the mongols, the higher the efficiency the more damage can be inflicted. 20% efficiency is lost for most actions.

Temperature
Temperature starts at 0. The more the player makes use of the laser and torpedos, the more the temperature gauge rises. When the temperature reaches a value of 100, the guns will overheat and become inactive until a player visits a starbase to repair up and cooldown. 20 temperature is lost for most actions.

Mongol Counter
The mongol counter will be randomly generated upon runtime. This will be a value from 5 to 10. This is the total of mongols that will be present in the game that the player needs to go and destroy in order to claim victory. The mongol counter will decrement after each mongol death, as well rendering one less mongol in the counter.

SCREENS / GAME STATES

Menu
Upon game start, the player will be shown the menu screen. Here they will be given a list of options to answer such as whether they would like to see a brief, what their name is, what difficulty they would like to set the game to and finally what their password will be. If the user states they would like to see a brief, a panel will appear which the player will be able to flick through with the right arrow key. Providing no to the question will close the brief if it is open and proceed to the next question. After providing a password the player will be taken to the game screen.

Game
Inside the game screen, the player will be greeted to a display of different panels. These include the Command, Action, Status, Systems, Controls and Report panels. The Action panels can be brought up after entering in the appropriate commands. The player will also see two grids. A small grid in the top left and a larger grid in the top right made up of a series of numbers. These are the sector and quadrant grids. The sector grid is the player's local position in the current quadrant, and the quadrant grid is the player's location in the galaxy.

Win
If the player successfully destroys all mongols, they will be greeted with the victory screen. Here a short animation will be played of the victory text enlarging from miniscule to full scale. The score they gained shall be shown on the line below.

Lose
If the player fails in their mission to destroy all mongols, they will be taken to the defeat screen. As with the victory screen, a short animation of the defeat text will be played and a line below it telling the player the score they achieved.

DEVELOPMENT PLAN

CREATE THE MENU STATE

First I will create an array of Texts and render them to screen. I will then create the input text and formulate a function to align the input text to the questions. To check which question the player is currently on, I will create an enum class called ActiveText and assign each constant to each question.

Then I will create a class for my panels. This will hopefully make it easy for me to initialise in each panel that I need, all created for me inside the panel class. I shall then use this class to create a panel for the brief.

CREATE THE WIN AND LOSE STATE

I want a text font that is big and bold so first I will locate the font I need and then load in the font. I will then initialise and render the victory text to screen, applying the new font to it. I shall do the same for the defeat text. Finally I will set up the score variable and render it underneath so that the player's score is shown at the end of the game.

CREATE THE GAME STATE

First I will lay out the different panels such as the status, systems, controls, report, and command panels. Then I will create a class for the different grids. Next the objects will be randomly positioned in the sector grid before then implementing player movement so that I can get some core functionality going. Proceeding this, I will implement the shooting mechanic. This will be similar code wise to the movement except instead of altering the players position it will generate a damage output and inflict this upon the mongols.

TECHNICAL DESIGN DOCUMENT

FEATURES

TYPING TEXT TO THE GUI

To get the text rendered on screen I will use a Text object. I will then create a string variable to store and capture key presses so that when a key press is made, each character is stored inside the string variable.

```
Text text
Std::string string
text.setString(text += key)
```

SPRITE POSITIONING

To set up the sprites spawn locations within the sector grid, I will use a for loop to generate a random number for all the sprites in the game. This random number will serve as the sprites positions in the sector grid. Each iteration of the for loop will set the sprites location and set that location to be occupied using a boolean. A while loop will be used to check each iteration to see if the random grid position generated is already occupied and if it is then it will generate a new number.

```
For all items in sprite
    Generate a random grid number
    while grid location is occupied
        Generate a new random grid number
    Sprite positioned in grid
    Sprite position set to occupied
```

STARSHIP CLASS

I will be creating a class to handle and store all of the ships functions in the game such as shooting, restoring ship health, taking damage, increasing / decreasing weaponry efficiency and increasing / decreasing weaponry temperature. These will be public functions that are accessible by all objects that inherit from it.

```
Starship : GameObject
Player : Starship
Enemy : Starship
```

SCENEMANAGER

I will be creating a class to provide all the necessary functionality required by the classes that inherit from it i.e. other scene classes such as the menu class. It will also handle the changing of the game state and also where the input text is positioned through the use of an active text enum class.

```
SceneManager
MenuScene : SceneManger
GameScene : SceneManger
WinScene : SceneManger
LoseScreen : SceneManger
```

GAMESTATE

The game state will be handled by an enum class. This will hold the states for the menu, game, win and lose state. I have chosen to use an enum class for its type safety and ease of readability. To make sure that i can alter the game state across classes so that it acts as one, i have created a function that returns a static instance of the game state.

```
enum class GameState { MENU, GAME, WIN, LOSE }
```

GRID CLASS

The sector and quadrant maps will be grids made up of a vector of unique pointer gameobjects. Making them unique pointers will handle the memory for me so that I don't have to manually delete the sprites when I'm done with them.

PLAYER CLASS AND ENEMY CLASS

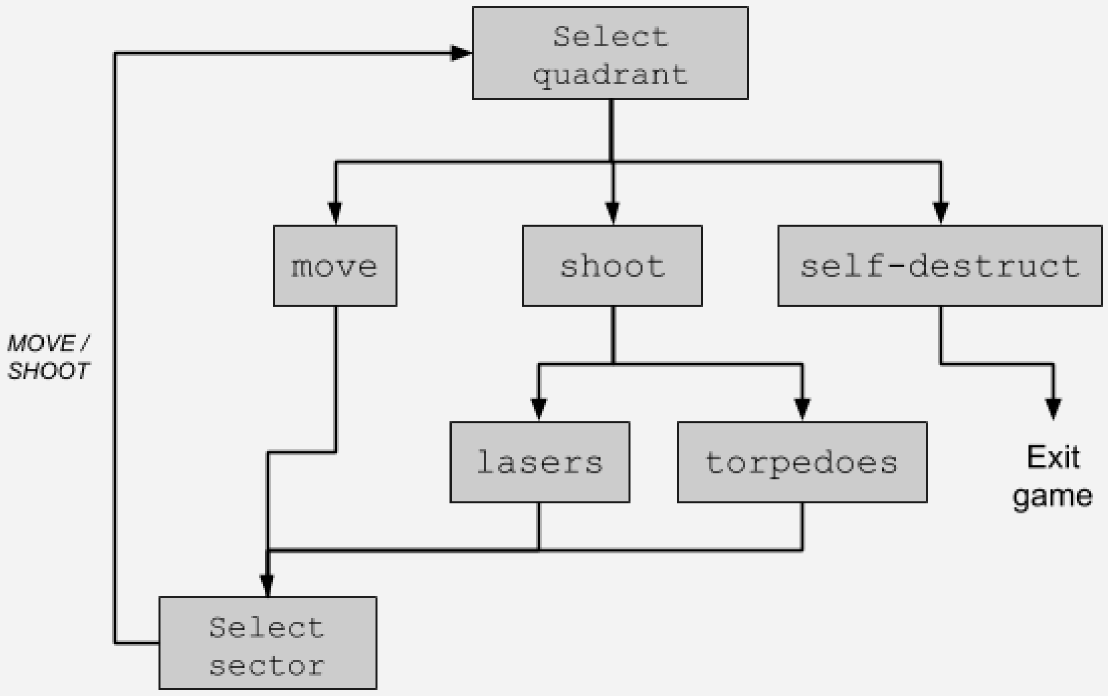
The player and enemies will also be made up of a vector of unique pointer gameobjects. Using a vector is preferable over an array as a vector allows me to dynamically grow the size at run time. These classes will store the functions required by the player and enemy classes such as setting and getting information about the objects' different attributes.

GAME ENGINE / FRAMEWORKS / LIBRARIES

I will be using the ASGE framework using modern C++ to design and build the game. I will also be using a range of libraries to provide some functionality required in my game such as the vector and random library, which will be used to randomly distribute the sprites across the sector, and also the standard template library (STL).

HIGH LEVEL DIAGRAMS

GAME LOOP


Select quadrant
move
shoot
self-destruct
MOVE / SHOOT
lasers
torpedoes
Exit game
Select sector