EOPSY Lab 5, 6, 7 project concept

Maciej Domański, Krzysztof Rudnicki, Gabriel Skowron-Rodriguez

May 12, 2022

1 Introduction

1.1 Goal of project

Creating an analogy in form of a real time strategy game to operating system processes and how they function.
Each soldier in our game will represent a process which can receive and send signals from and to "the general" (player), soldiers communicate with each other, exchange information, divide resources among themselves and work together to achieve a goal or set of goals.
We will focus on the parts of the game which represent how processes work in operating systems first, then on how the game actually looks and works

1.2 Tools

We will use the Unity Game Engine, for simplicity's sake we will use Monobehaviours on prefabs to represent the soldiers. This is not the most performant solution but it is easy enough to implement in realistic (one month) timeframe. When it comes to more detailed usage of different tools to accomplish different goals all of them will come up as the project progresses and we actually work on the tasks.

1.3 Dividing Tasks

Every week our group will meet on Monday at 12:00 and work for approximately 3 hours. This will ensure that everybody does a similar amount of work.
Task division will be done using Trello before or at the beginning of the weekly meeting.

1.4 Workflow

We use gitlab hosted on studia elka servers. There are 3 types of branches in our project:

1. Main - Rarely used, hosting stable versions of the game
2. Development - Branch where we merge all individual branches and actively change on weekly basis.
3. Individual branches - Created for each task and merged when this task is finished

We also use separate scenes on which we will work for each member of the project.
Branching and separate scenes minimize the risk of corrupting the gitlab project and simplifies merging and management of the project.

2 Goals

2.1 Minimum Viable Product

This is a minimum goal we want to achieve in order to consider this project as finished

Goal of the game Destroying enemy base. For simplicity's sake, the enemy base is just a "soldier" without the ability to attack or move, with more health points. "Base destruction" is accomplished by reducing its health points to zero.

Realization of the goal Attacking the base with our soldiers.

Map Map is divided into tiles

2.1.1 Soldiers

Exactly one type of soldiers
Exactly one squad
Exactly one formation (Rectangle or any other that we find the easiest to implement)
New soldiers spawn (appear) every x seconds in the squad
Soldiers have following attributes:

- Health Points
- Range of Attack
- Range of View

- Damage per attack
- Speed of attack

We have decided to split range of attack and range of view since combining them may lead to difficult code problems to solve should we reach higher level(s) of our project.
Soldiers occupy physical space - a tile

2.1.2 Soldiers Communication

Soldiers receive global orders from the player but they use their local vision to fulfill those orders. (For example by looking at where their neighbours stand and adjust their position accordingly). Used to simulate communication between operating system and processes.
In order to communicate something to the whole squad (like killing of an enemy soldier), the soldiers sends a message to its neighbours, causing the info to be distributed to the whole squad immediately
This is used to simulate processes communicating with each other.
Each soldier has its own Action "stack". Every tick a single (topmost) action from the Soldier's Action stack is executed
There are process interrupts sent by the player which force the soldier to perform a special action instead of executing what is on the action stack.

2.2 Squads

Exactly one squad on map

Listing 1: Pseudo-code for Squad class

```
class Squad{
    List<Soldier> soldiers
    Queue<Order> orders
}
```

2.3 Fighting

If an enemy is within a Soldier's attack range, the Soldier will perform an "attack" action
"Attack" means that the closest enemy to the soldier, receives damage (which means that its health gets decreased, accordingly to the soldier stats)

2.3.1 Player

Player orders are first in priority for soldiers, (like interrupt signals)
Player sees everything on map

2.3.2 Enemy

One stationary squad of enemy soldiers
Enemy base
No AI

2.4 Expansion 1

This is what we would like to introduce should we finish MVP. We mostly focused on operating systems and processes related features.
Expansion 1 means that those are first priority features we would like to implement after finishing minimum viable product.

2.4.1 Fighting

Priority points determining who the soldiers should attack based on distance from the enemy, how many friendly soldiers are nearby or the enemy health.
Soldiers automatically get in fighting distance to enemy soldiers
When soldier sees enemy he communicates so to the squad and the whole squad goes towards enemy

2.4.2 Squads

More than one squad
We can transfer soldiers between squads

2.4.3 Commander panel

Commander panel with all info regarding soldiers and squads, amount of enemy soldiers killed, where they are, whether they exist, ability to remove squads which do not send those statistics.
If there is at least one living soldier in a squad he will send every x ticks statistics:

- How many soldiers are in the squad
- Squad position
- How many enemy soldiers does he see

2.4.4 Player

Symbolically player sees different color outlines of the squads based on whether they send statistics or not

2.5 Expansion 2

Expansion 2 means that those are second priority features we would like to implement after finishing minimum viable product AND expansion 1 Creating soldiers in the main base and spawning them near the base in a new squad.
Soldiers have limited resources (as in limited memory and CPU power)
More soldier types with different communication and attributes
Player sees exactly and only what the soldiers send him.
Resources with stable generation of x resources per minute
Resource points to make generation faster
Physical, realistic placement of soldiers on map instead of tiles