Università degli Studi di Pavia

Bachelor degree in Artificial Intelligence

Caste wars — Final exam project — A.Y. 2021/22

Stefano Ferrari

Computer programming, Algorithms and Data str., Mod. 1

Contents

date: January 11, 2022
version: 0.9

The final exam project consists in the design and implementation of a strategy game named "Castles war". The game is intended to be a 2D side-view game, implemented using the library `pygame`.

1. Rules

The game is played by two players, each of which manages a castle. The castle has a wall that protects the inside buildings and units. The goal of the game is to destroy the opponent wall (and hence conquer the castle).

Each castle has the followig buildings:

- a tower, which can hit the attackers
- a barracks, which can train units
- a mine, from which workers can extract resources

Each player starts the game with a population of one unit of each type and an initial amount of resources (`INIT_RESOURCES`).

The game is turn-based, although the turn can be so short in time that the rules can be adapted to a real-time game.

The game world is characterized by the following measurement units:

Time The time is measured in *turns*.

Distance The distance is measured in *steps*. For graphical purposes, the equivalence between steps and pixels can be considered, for the sake of simplicity.

Health Each element in the game has a property called *health* measured in *health-points* (HP). An element can have an active role if its health is larger than 0.

Each turn consists in:

command in which the player can issue an order of the following:

- train unit;
- dispatch unit;
- unleash all the military units;

combat in which the elements of the game are updated using the rules specified in the following;

work in which the buildings properties are updated using the rules specified in the following.

2. Units

There are three types of units:

Worker Civil unit that can dispatched in mine to extract resources or to the wall to restore it.

Swordsman Military unit that can attack adversary units and wall with a short range attack.

Archer Military unit that can attack adversary units and wall with a long range attack.

Each unit has a behaviour and ability ruled by its parameters, further explained in the following.

2.1 Worker

A new worker can be generated by the barracks using `WORKER_COST` resources. The generation of a worker requires `WORKER_TRAIN` turns to be completed Once created, the worker wait in the barrack until it is dispatched to the mine or to the wall. The worker moves at a speed of `WORKER_SPEED`. When it is working in the mine it generates `WORKER_PROD` resources per turn. When the worker is at the walls, it can restore `WORKER_REPAIR` HP of the wall per turn.

2.2 Swordsman

A new swordsman can be generated by the barracks using `SWORD_COST` resources. The generation of a swordsman requires `SWORD_TRAIN` turns to be completed Once created, the swordsman wait in the barrack until it is released to attack the adversary castle. The swordsman moves at a speed of `SWORD_SPEED` until it reaches an adversary unit (or the wall) in its range, i.e., when the distance from the adversary unit is no larger than `SWORD_RANGE`. During the combat, a swordsman delivers a damage of `SWORD_HIT` to the target. After having hit a target, the swordsman is inactive for `SWORD_REST` turns. During the combat, the swordsman can also receive a damage that will decrease its health, which is initially `SWORD_HEALTH` HP.

2.3 Archer

A new archer can be generated by the barracks using `ARCHER_COST` resources. The generation of an archer requires `ARCHER_TRAIN` turns to be completed Once created, the archer wait in the barrack until it is released to attack the adversary castle. The archer moves at a speed of `ARCHER_SPEED` until it reaches an adversary unit (or the wall) in its range, i.e., when the distance from the adversary unit is no larger than `ARCHER_RANGE`. During the combat, an archer shoots an arrow which delivers a damage of `ARCHER_HIT` to the target. The arrow travels at a speed of `ARROW_SPEED` until it reaches the adversary (or go out the game world). After having shoot an arrow, the archer is inactive for `ARCHER_REST` turns. During the combat, the archer can also receive a damage that will decrease its health, which is initially `ARCHER_HEALTH` HP.

3. Buildings

The buildings in the castle are the following:

Wall It is the protection of the castle. Initially, it has a maximum health of `WALL_HEALTH`, that can decrease when hit by an adversary unit. The wall health can be restored by the workers

up to `WALL_HEALTH`. When the wall health reaches 0, it is destroyed and the game is over. The winner is the adversary (unless in the rare case in which also the adversary wall is destroyed in the same turn).

Barracks It can train new units and hold them until a dispatching command is given.

Mine It allows to obtain resources through the activity of the workers.

Tower The default defense of the castle. It can shoot arrows to the adversary units closer than `TOWER_RANGE`. During the combat, the tower shoots an arrow which delivers a damage of `TOWER_HIT` to the target. The arrow travels at a speed of `ARROW_SPEED` until it reaches the adversary (or the ground). After having shoot an arrow, the tower is inactive for `TOWER_REST` turns.

4. Arrows

The arrows are intended to travel using a linear trajectory. Hence, the arrows shoot bt archers will travel horizontally, while the arrows shoot by the towers will travel diagonally (from the tip of the tower to the ground).

5. Commands

Each player can control the game through the following actions:

Train This command is issued to the barracks in order to start the generation of a new unit of a given type. If the player has enough resources to the aim, the resources are decreased by the amount needed and the train can start. Otherwise, the command in ignored (and eventually reported).

The commands are issued by pressing the following keys:

	Player 1	Player 2
Worker	`'q'`	`'p'`
Swordsman	`'w'`	`'o'`
Archer	`'e'`	`'i'`

Dispatch This command is issued to change the action of a unit. When a worker is dispatched to a building (mine or wall), the command is issued first to the workers in the barracks. If no workers in barracks are available, a worker availability in the other building is checked. If positive, the worker is moved in the target building, otherwise the command is ignored (and eventually reported). Similarly for the military units, their availability is checked and in case the order is fulfilled, otherwise the command is ignored. The unleash all command is applied to all the military units in the barracks, which are sent to combat.

The commands are issued by pressing the following keys:

	Player 1	Player 2
Worker to mine	`'a'`	`'l'`
Worker to wall	`'s'`	`'k'`
Swordsman attack	`'d'`	`'j'`
Archer attack	`'f'`	`'h'`
Unleash all	`'z'`	`'m'`

Pause The Pause command temporary stops the execution of the game. The command is issued by pressing the space bar. The game can continue when the space bar is pressed again. Both the command can be issued by any of the players.

Save The Save command store on a file the game status, using the format given in Sect. 7..

Load The Load command read the initial game status, using the format given in Sect. 7.. The game will start in a Pause status and the operation will start when the space bar is pressed.

6. Constants

6.1 Game constants

- `SCREEN_WIDTH`:
- `SCREEN_HEIGHT`:
- `BORDER`:
- `BUILDING_SEP`:
- `WALL_POS`:
- `WALL_WIDTH`:
- `MINE_POS`:
- `MINE_WIDTH`:
- `BARRACKS_POS`:
- `BARRACKS_WIDTH`:

6.2 Castle constants

- `INIT_RESOURCES`: initial amount of resources

6.3 Units constants

Worker:

- `WORKER_COST`: amount of resources to generate a new worker
- `WORKER_TRAIN`: number of turns required to generate a new worker
- `WORKER_PROD`: amount of resources that a worker generates each turn it spends in mine
- `WORKER_SPEED`: distance covered by a worker in one turn
- `WORKER_REPAIR`: amount of wall HP restored in one turn

Swordsman:

- `SWORD_COST`: amount of resources to generate a new swordsman

- `SWORD_TRAIN`: number of turns required to generate a new swordsman
- `SWORD_SPEED`: distance covered by a swordsman in one turn
- `SWORD_RANGE`: maximum distance at which the swordsman can hit an adversary unit
- `SWORD_HIT`: damage in HP delivered to the target
- `SWORD_REST`: turns of inactivity after delivering a damage
- `SWORD_HEALTH`: amount of HP initially given to the swordsman

Archer:

- `ARCHER_COST`: amount of resources to generate a new archer
- `ARCHER_TRAIN`: number of turns required to generate a new archer
- `ARCHER_SPEED`: distance covered by an archer in one turn
- `ARCHER_RANGE`: maximum distance at which the archer can target an adversary unit
- `ARCHER_HIT`: damage in HP delivered to the target by an arrow shoot by an archer
- `ARCHER_REST`: turns of inactivity after shooting an arrow
- `ARCHER_HEALTH`: amount of HP initially given to the archer

Arrow:

- `ARROW_SPEED`: distance covered by an arrow in one turn

6.4 Buildings constants

Wall:

- `WALL_HEALTH`: maximum of HP of the wall

Tower:

- `TOWER_RANGE`: maximum distance at which the tower can target an adversary unit
- `TOWER_HIT`: damage in HP delivered to the target by an arrow shoot by the tower
- `TOWER_REST`: turns of inactivity after shooting an arrow

7. Formats

7.1 Save/Load

The game status consists in the value of the properties of all the element of the game.

The following textual format is proposed. The file is structured in three macroblocks, corresponding to the properties of the game, the two players. The blocks are limited by `START` and `END` keywords, followed by `Game`, `Player1`, and `Player2` keywords. Inside the blocks, there is the list of the properties (name and value) of the elements, one per line. For the game, the properties can be:

- `TURN`: #turns

For players:

- `RESOURCES`; #resources
- `WALL`: current wall health
- `WORKER`: worker position (`BARRACKS`, `WALL`, `MINE`)
- `SWORDSMAN`:
  - `POS`: swordsman's position (coordinate in the game world)
  - `HEALTH`: swordsman's health
- `ARCHER`:
  - `POS`: archer's position (coordinate in the game world)
  - `HEALTH`: archer's health
- `ARCHER_ARROW`: arrow position (coordinate in the game world)
- `TOWER_ARROW`: arrow position (coordinate in the game world)

Notice that the position of all the elements (units and archer arrows) are monodimensional (only its horizontal coordinate is important), while for tower arrows position two coordinates are needed.
Example:

```
START Game
TURN 4322
END Game

START Player1
WALL 1000
END Player1

START Player2
WALL 892
ARCHER POS 126 HEALTH 20
TOWER_ARROW 800 32
END Player2
```

8. Resources

Pygame docs: `https://www.pygame.org/docs/`
Sprites: `https://opengameart.org/content/hack-and-slash-2d-avatars`