INFO0004-2: Project 3
PACoronam (PAC-MAN 2020 edition)

April 21, 2020

1 Introduction

PAC-MAN is a fairly simply game: the hero (PAC-MAN) lives in a no-exit maze full of PAC-MAN food, with 4 hungry PAC-MAN-eating monsters. The goal is for the player, using the 4 directional arrow keys on the keyboard, to steer PAC-MAN to eat all of its beloved food before being caught and devoured by a monster.

Note that the monsters do not eat PAC-MAN 's food, although, as we will see later, when you are not playing and they get hungry, they have a pretty bizarre diet.

In this project, you will be building a PAC-MAN game in C++14, using the SFML 2.5.1 library (Simple and Fast Multimedia Library). This library provides easy windowing and user input, timing and graphics rendering facilities.

Your game must render at 60 frames (images) per second.

2 PAC-MAN's world (a.k.a. the maze)

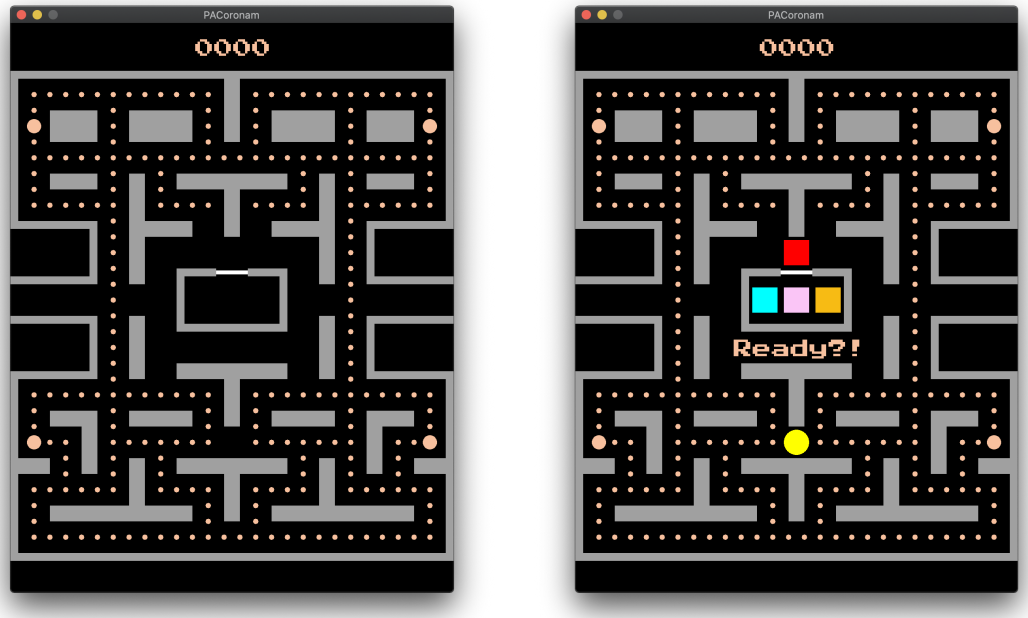
The maze is depicted in figure 1a. In this figure, you can see grey walls, as well as PAC-MAN's food (the dots). Obviously, no character in the game can go through walls (don't believe people who wrongly tell you that PAC-MAN is chased by ghosts!). In other words, any tile that contains a bit of wall is inaccessible.

All in all, there are 244 dots for PAC-MAN to eat. There are 240 small dots, or *treats*, which PAC-MAN eats because it likes them and they earn points (see section 6: "Scoring"). The 4 bigger dots, or *power pills*, are very special: they momentarily turn PAC-MAN into an invincible monster-gobbling machine (thus turning the tables on the monsters).

The area in the center is the *monster house* (with a white door), which PAC-MAN can never enter. You can also see that there are six areas of the world that are completely inaccessible.

You may also have noticed that half way up the maze, there are what look like two *exits*, one to the right, and one to the left. These *exits* are in fact connected by a high-speed underground tunnel, so that any characters (PAC-MAN or monsters) exiting on one side soon reappears on the other side, after a brief delay through the tunnel.

Figure 1b shows the configuration of the maze at the start of the game.



(a) Maze with food. **(b)** Start configuration.

Figure 1: ***PAC-MAN's world.***

3 Behaviours

All the characters move smoothly (*i.e.* with a continuous movement) around the maze. Also, with a single exception discussed below, they always stay in the middle of the hallways.

Although the characters move smoothly in the maze, a lot of the game logic is *grid-based*. To support this logic, a grid of 36 rows and 28 columns is super-imposed on the world, making a total of 1008 grid tiles (see figure 2a). A character is considered to occupy a tile when its center is within the tile.

There is a single *reference speed* used to compute the actual speed of the various characters at different moments in the game.

3.1 PAC-MAN

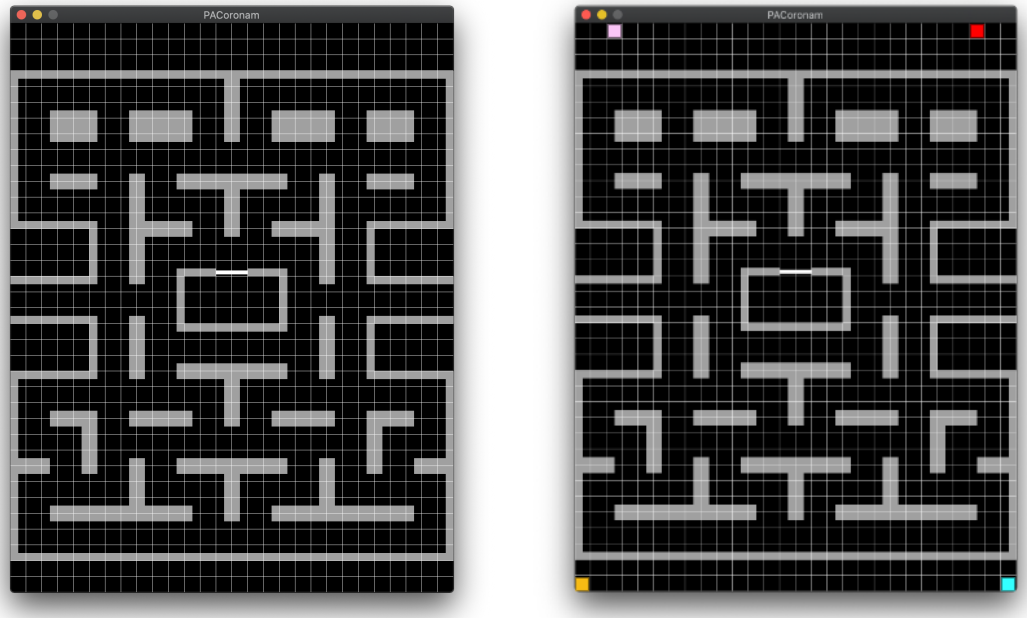
PAC-MAN normally moves around the maze at full reference speed. The only exception to this rule is after eating a power pill, when it enjoys a momentary 5% speed boost.

Whenever possible, PAC-MAN simply obeys the players directional instructions.

Once in movement, PAC-MAN will only stop when it encounters a wall in its direction of movement.

3.2 Cornering

When PAC-MAN is on a tile where it can take a direction orthogonal to its current direction of travel, something special happens if it receives an order to take that orthogonal direction while it



(a) Logical grid. **(b)** Scatter mode targets (home tiles).

Figure 2: ***Game logic grid.***

has not reached the middle point of the tile yet: it will immediately start moving at a 45 degree angle towards the wanted direction. PAC-MAN will keep that bearing until it reaches the middle of the hallway it is going into, at which point it resumes a vertical or horizontal movement. Figure 3a shows a trace of such movement dynamics while changing the direction of movement from the *east* to the *north*.

This is the only exception to the strictly *middle-of-hallway*, *parallel-to-axis* rule for movements otherwise imposed to all characters. This way to *cut the corner* can give PAC-MAN much needed breathing space when monsters are in hot pursuit, as monsters can only take right angle turns from the middle of a tile: PAC-MAN turns can be shorter than monster turns.

Note that while PAC-MAN is taking a turn in such a way, it is committed to that turn and must no longer obey change of direction orders, until it has reached the middle of the hallway it is heading into.

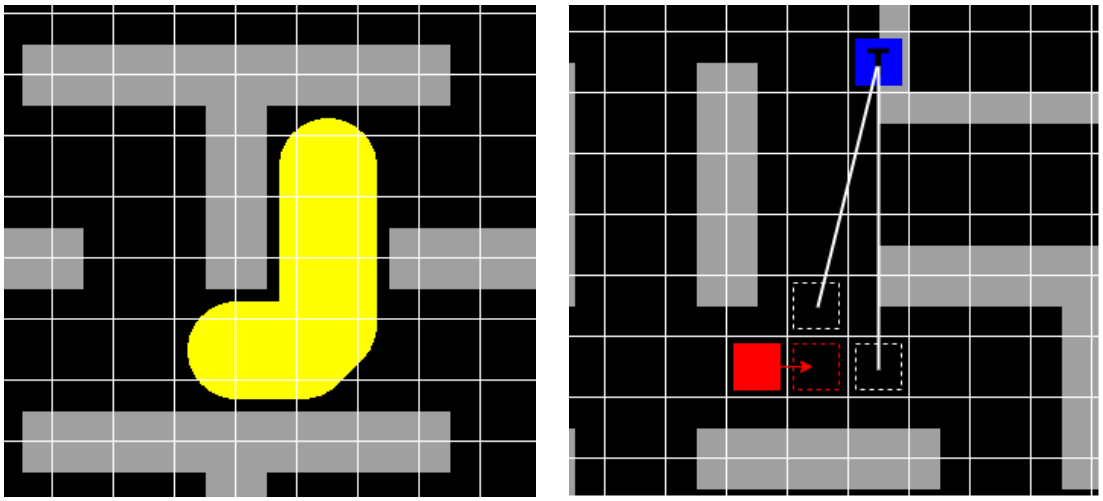
If PAC-MAN is exactly on the middle point of its grid tile when it receives an orthogonal turn order, it immediately takes a right angle turn in the wanted direction.

If PAC-MAN is past the middle point of its tile direction, it has missed the turn and ignores the orthogonal turn order.

Finally, as soon as PAC-MAN reaches a grid tile containing food, it eats it. But when it has eaten, PAC-MAN cannot move for the duration of one whole frame, while it is digesting.

3.3 The monsters

Most of the time, the monsters move at a speed equal to 95% of the reference speed.



(a) Cutting the corner. **(b)** Routing.

Figure 3: ***Junction behaviour.***

Routing

When they are out of the monster house, the monsters have, at all times, a grid tile they want to reach. Exactly which tile depends on several factors (see below). Nevertheless, with a target tile in mind, a monster makes *routing* decisions by looking one tile ahead of its current tile in its direction of travel. It then considers all the neighbouring tiles of that tile in turn, except for the tile it is currently on. Using those tiles, it will decide on its next move when it gets in the middle of the tile ahead: it will turn in the direction of the neighbour tile that gets it the closest to its target tile, with distances measured as euclidean distances. If there are some ties (because neighbour tiles are at equal distance from the target tile), these are broken by picking a direction in the following preferred order: *north*, *east*, *south*, *west* (or clockwise priority).

It is probably easier if monsters make routing decisions when they reach the middle of a tile.

The routing is illustrated in figure 3b: the (red) monster is travelling east and looks ahead to the tile marked with a dashed red square. The possible neighbours of the red dashed square tile are marked with white dashed squares. Measuring the distances between these white dashed square tiles and the monsters target tile (marked with a blue square containing the letter T), the monster will decide to turn north when it reaches the red dashed square tile, because it is closer, as the bird flies, to the target tile.

Modes of behaviour

Monsters have three modes of behaviour:

1. *Chase mode*, where the monsters chase down PAC-MAN, more or less aggressively.
2. *Scatter mode*, where the monsters disperse towards the corners of the maze.
3. *Panic mode*, that the monsters enter when PAC-MAN eats a power pill.

While in play, a pattern of 9 seconds of scatter mode followed by 30 seconds of chase mode is repeated indefinitely.

When PAC-MAN eats a power pill, the monsters enter panic mode for 7 seconds. They all turn blue for the first 6.5 seconds, then turn white for 0.5 second, to signal the imminent end of PAC-MAN's

powers.

Note that panic mode does not change the timing of the scatter/chase pattern: panic mode simply overrides, temporarily, the prescribed behaviour of the monsters in the scatter and chase modes[1].

In panic mode, the speed of the monsters is reduced to 77% of the reference speed, and the monsters simply take random turn decisions at junctions (but they cannot do u-turns).

The only difference between scatter and chase mode is how the monsters pick their target tiles. In scatter mode, each monster has a predefined home target tile, shown in figure 2b.

In chase mode, each monster has its own way to pick a target tile. It is time to be introduced to these monsters!

Shadow, a.k.a. *Blinky*, the red monster. He is the most aggressive of the monsters in chase mode, because it always picks PAC-MAN's grid tile as its target tile. As you will have noticed on figure 1b, Blinky starts the game outside the monster house. Blinky is a blood thirsty, PAC-MAN hungry bastard, so he *starts the game in chase mode*, giving him a full 39 seconds for harassing PAC-MAN, before entering scatter mode for the first time. Blinky always starts heading west as its first move.

Speedy, a.k.a. *Pinky*, the pink monster (who would have thought?!). In chase mode, Pinky is a crafty little bugger who will try to ambush PAC-MAN by looking at PAC-MAN's grid tile and PAC-MAN's direction of travel, and picking as its target tile the tile exactly 4 tiles in front of PAC-MAN. Pinky starts the game inside the monster house, just behind the door, and comes into play as soon as the game starts. Like *all the monsters*, Pinky goes west after exiting the monster house.

Bashful, a.k.a. *Inky*, the blue-green (its colour is cyan, actually) monster. Inky is the third monster to join the game, exiting the monster house when PAC-MAN has eaten 30 pieces of food, or 15 seconds, whichever comes first. In chase mode, Inky considers the vector from Blinky's grid tile to the tile exactly 2 tiles in front of PAC-MAN, doubles that vector, and whatever grid tile the extremity of that vector falls, is its target tile.

Pokey, a.k.a. *Clyde*, the orangey monster. Clyde will join the game after PAC-MAN has eaten one third of its food (that's 82 pieces of food) or 60 seconds, whichever comes first. Clyde is a bit loopy, and looks like it cannot make its mind up: in chase mode, if Clyde is closer than 8 grid tiles to PAC-MAN, Clyde will choose its home tile as its target (just like in scatter mode), but if Clyde is further than 8 tiles from PAC-MAN, its target tile is PAC-MAN's tile (just like what Blinky does). In other words, if you imagine a circle with a radius of 8 tiles around PAC-MAN, Clyde flees towards its home tile when within the circle, and goes straight for PAC-MAN when out of the circle.

It is worth noting that target tiles can fall outside the screen.

When leaving the monster house, all monsters go west, and (re-)enter the fray in the mode dictated by the global scatter/chase cycle.

We said that while routing, monsters cannot reverse direction (make u-turns). However the monsters *must* reverse their direction of travel, when the mode changes from scatter to chase, scatter to panic, chase to scatter, and chase to panic. These reversal of direction serves to visually signal the change of mode to the player. Note that when the change of mode is signalled, the monster will only actuate this change upon their next routing decision (ignoring whatever routing decisions they may have planned for that tile).

[1] For instance, if a chase to scatter mode occurs during panic mode, the monsters go into scatter mode at the end of the panic.

4 Collision detection

You need a mechanism to detect collisions between PAC-MAN and the monsters. Note that there is no collision detection between the monsters.

The collision detection mechanism used in this game is rather primitive: a collision is declared if the tile occupied by PAC-MAN and the tile occupied by a monster is the same (*i.e.* they both have their center in the tile).

If there is a collision between PAC-MAN and a monster during either scatter or chase mode, PAC-MAN gets gobbled up by the monster and the game is finished with a loss for the player (in this version, PAC-MAN only has one life). On the other hand, if there is a collision during panic mode, than it is the monster that gets eaten.

When a monster is eaten, its becomes somewhat transparent, and needs to go back to the monster house to be regenerated. An eaten monster is technically momentarily out of the game, and travels at a speed which is 130% of the reference speed. It tries to reach the house using its normal routing, and gets regenerated as soon as it reaches Pinky's starting position inside the house. Once regenerated, the monster comes back out of the house at the normal speed (turning left, in whatever mode the scatter/chase cycle is in).

Note that a monster that has just been eaten *can* reverse its direction of travel immediately, if turning onto itself is the best local move towards the house.

This simple collision detection mechanism has certain limits: PAC-MAN and monsters will touch all the time without consequences: this because, as we will discuss later, the characters are, on screen, bigger than a tile on screen. Worse, with the right timing, a monster and PAC-MAN can swap tile, passing over each other without triggering a collision: this is not a bug, it is a documented feature.

5 Restrictions

Whenever an in-play (*i.e.* not eaten) monster enters one of the red areas depicted in figure 4a, its speed is reduced to 55% of the reference speed: monsters are really slow there!

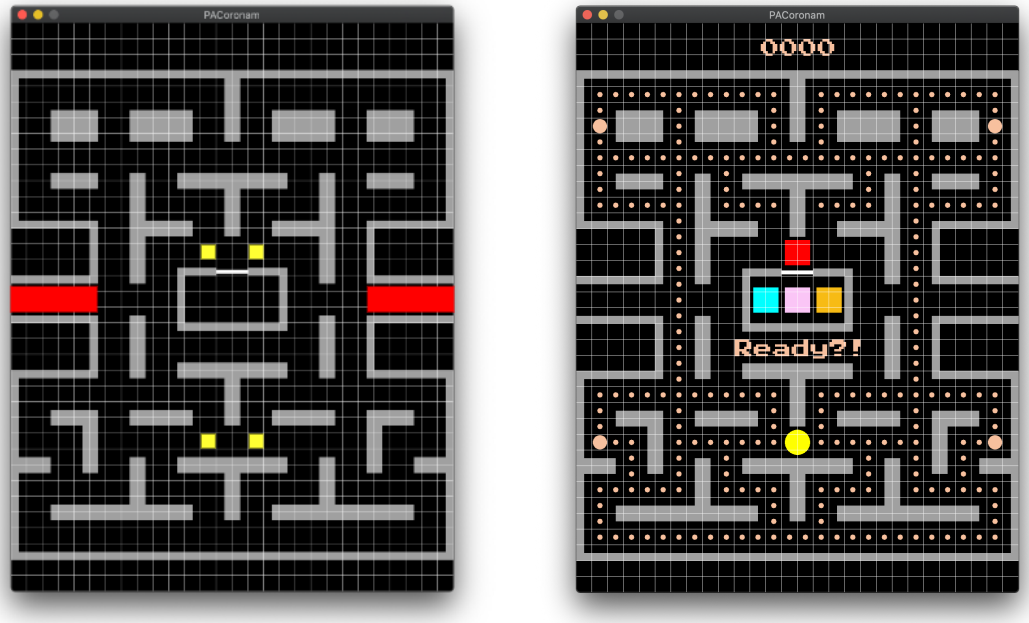
In figure 4a, you will also have noticed 4 junction tiles marked yellow. In scatter and chase mode, the north turn at those tiles is unavailable to monsters: these are good places for PAC-MAN to get rid of chasing monsters!

After a character exits the maze on one side, it will re-appear on the other side after a brief moment. You can calibrate the time the characters pass in the underground tunnel.

But there is one more thing about the underground tunnel: while nobody, apart from PAC-MAN and the monsters, really knows what happens in there, we do know that collisions never happen in the tunnel. No risky business there!

6 Scoring

The player earns 10 points whenever PAC-MAN eats a treat, and 50 points when it eats a power pill. While under the power of a pill, PAC-MAN also gets 100 points for eating the first monster. The number of points for eating each subsequent monster (under the power of the same pill) is double the points earned from the previous monster. If eating a new power pill *extends* the panic the previous pill triggered, this doubling of points is carried onto the new (extended) panic period (which will finish 7 seconds later).



(a) Restrictions. **(b)** Writings.

Figure 4: ***Restrictions for monsters and writing placement.***

7 Strange times

Legend has it that when you are not playing PAC-MAN, the hungry monsters survive by eating wild animals. As you should all know by now, this is a filthy habit that can cause serious trouble.

And sure enough, at a random time within the first 60 seconds of play, a random monster will become infected with some strange virus.

Once infected, a character will incubate the disease (and show no sign that anything is wrong), but will be contagious, for 20 seconds. After this incubation period, the character will be sick for 10 seconds. A sick character turns pale green, and its speed is overridden to 20% of the reference speed. Note however that while a character's colour and speed are changed by the disease, their behaviour is nonetheless unchanged: monsters keeps chasing, scattering and panicking, while PAC-MAN keeps responding to the player's command and eating. It is just that when sick, they do all that at 20% reference speed.

Monsters are filthy little pricks, and while they do not like, at all, eating PAC-MAN's food, they keep fiddling with it when they are on a tile containing food. And of course, when they are incubating or sick, that contaminates the food: any viral load stays active on the food for 6 seconds. That's how the virus can pass from monster to monster! Note that once infected, a character cannot be re-infected (*i.e.* the incubation timer is not restarted when an infected character is exposed to a new viral load).

On the other hand, poor PAC-MAN, who is the victim in all this, can pick up the virus by eating, either a contaminated piece of food, or a panicked, infected or sick monster.

But there is good news: everyone eventually recovers from the disease (after the 10 seconds sickness period). And once recovered, the characters are immune to new infections: if the player manages to make sure PAC-MAN is not eaten while weakened by the disease, he/she can forget

about new pandemics.

Note that eaten monsters are always regenerated with a new immune system, and therefore loose their potential immunity to the disease. Note also that an eaten monster, on its way back to the monster house, does not infect anything or anybody.

8 Looks

In this version of the game, you will opt for a slick, simple look for the characters:

- PAC-MAN is a yellow circle, with a radius of 0.8 times the size of the side of a tile.
- Each monster is a square, whose side is the size of 1.6 times the size of a tile.

No animation, no sound. Sorry if you feel let down by the lack of *wacka, wacka, wacka...*

Blinky is red, Inky is cyan. Pinky is RGB(250, 197, 246), Clyde is RGB(247, 187, 20). A panicked monster is blue (then briefly white), and an eaten monster is RGBA(0, 0, 255, 127).

A sick character is marked, in its center, with a circle of radius half of the size of a tile side, and colour RGB(206, 252, 192).

The background of the maze is black, the walls are RGB(160, 160, 160), and the food and writings are RGB(247, 192, 158).

A treat is a circle of radius tile size divided by 6, and a power pill is a circle of radius half of (tile size minus 4).

Display the characters in this order: PAC-MAN, Blinky, Pinky, Inky and then Clyde.

For the side of a grid tile use a size that fits your screen resolution. Specify that size (in pixels) as the `CELL_SIZE` constant in a file named `constants.hpp`, at the top level directory of your project (same directory as your `main.cpp`).

The reference speed is 5 tiles per second.

We will supply a font file for all writings. Use a font size of 48 for all writings. Place the score, and all messages to the player, as shown in figure 4b. Display the message `Too bad!` when the player looses, and the message `You win!` when the player wins the game.

9 Remarks

- **Time keeping:** time keeping in the game does not need to be super precise. At 60 frames per second, each frame is just over 16 ms. This is plenty a resolution for your time keeping. Also, 16 ms is plenty of time to compute a frame of the game in C++, so you can just assume that the frames arrive bang on time, and you can use these frames as a time reference (you do not need a wall clock). Any clock drift from this assumption will not cause any problem.
- **Frame computation:** when computing a new frame, you should first update all the characters before checking for collision.
- **Starting the game:** the game starts when the player uses one of the directional arrow keys.
- **Be careful:** this project is fun, but contains many rules. Do spend some times making sure you understand all of them, so you can plan and design your program appropriately.
- **Game controls** will be a lot better if your code uses *raw keyboard access*, as opposed to the event queue, for the four directional arrows (see the SFML tutorial).

- Your game should include a reset functionality, which resets the game to the initial start state, at any time, when the player presses the `R` key.
- **`constants.hpp`** must also include, apart from `CELL_SIZE`, a constant `LATEST_VIRUS` that indicates the upper bound, in seconds, of the virus outbreak interval (default at 60), as well as a `DEBUG` constant, which, when set to true, shows visually (as in the demo videos), the evolution of the virus outbreak, as well as the grid.

9.1 Readability

Your code must be readable:

- Make the organisation of your code as obvious as possible.
- Use descriptive names.
- Complement your self-documenting code with comments, where appropriate.
- Choose a coding convention, and stick to it. *Consistency* is key.

9.2 Robustness

Your code must be robust. The `const` keyword must be used correctly, sensitive variables must be correctly protected, memory must be managed appropriately, the program must run to completion **without crash**.

9.3 Warnings

Your code must compile **without error or warning** with `g++ -std=c++14 -Wall` on the provided virtual machine (see eCampus). However, we advise you to check your code also with `clang++ -std=c++14 -Wall -Wextra -pedantic`.

9.4 Object-oriented approach

Remember this is an *object-oriented* course. Try to apply the object-oriented concepts where it makes sense.

9.5 Makefile

You **must** provide a `Makefile` that can build your game in the given VM. That `Makefile` should support separate, incremental compilation (*i.e.* if one file is modified, only that file and files depending on it should be recompiled).

9.6 Evaluation

Your code will be evaluated based on all the above criteria. Failure to comply with any of the points mentioned **bold face** can result in a mark of zero!

10 Submission

Projects must be submitted through the submission platform before **May 15th, 23:59 CEST**. Late submissions will be accepted, but will receive a penalty of $2^n - 1$ points (/20), where n is the number of days after the deadline (each day start counting as a full day).

You will submit a `s<ID>.tar.xz` archive of a `s<ID>` folder containing your C++ source code, and all other files needed for your code to run, and a `Makefile` to compile it (with a simple `make`), where `s<ID>` is your group ID.

This archive must also contain a report.pdf file, no longer than 5 pages, that describes the design and architecture of your code.

As this application is a game, the submission platform will not do basic checks on your submission. You can submit multiple times.