# C++ Project

## Arcade

Koalab koala@epitech.eu

*Résumé: In this project, you will have to create a gaming platform. This platform have to include at least two games and must be able to run with three differents GUI. Games and GUIs will be in dynamic loaded libraries. You have to choose between a little set of games and another little set of graphic libraries.*

# Table des matières

# I Nibbler, Snake, Blockade

Nibbler is a small arcade video game released in 1982. Its famous concept is mainly known throught the uberly cult game Snake. Nibbler was itself inspired by another great classic : Blockage, which also inspired Tron Light Cycle which is almost a clone of it.

The simplicity and the addictiveness of Snake make it available on almost every existing platform under various names.

For the old young, Snake means hours of procrastination in high school on old Nokia 3310 phones. For younger people, Snake is a tech2 project referring to a certainly underground pre-historic game, which interest only the old fossil pretending to be in the trend of their time.

It is never too late to discover this classical of video game : http://jeux-flash.jeu-gratuit.net/jeux_classiques/snake_250.html

As you can see, Snake is about moving a snake within a plan. This snake is represented by sections and it have to eat food, so that it can grow by a section each time it gets food. The game is over when the head of the snake hit an edge of the plan or one of the section. The goal of the game is to obtain a snake as long as possible.

Several versions of Snake exists. Some of them include obstacles, others have a score system, or bonuses etc.

## II Pacman

`Pacman` is an arcade video game from 1980. Its principle is to explore a labyrinth in order to eat all “pacgums” it contains while dodging ghosts. There is one “pacgum” by legal position. To win, `Pacman` must eat all of them.

A few “pacgum” allow the player to invert role : `Pacman` can, during a short period of time eat ghost instead of being eaten. Eaten ghost do not disappeir : their eyes must go back to a unaccessible (for the player) zone in the middle of the screen. They change to plain ghost after a short time.

http://jeux-flash.jeu-gratuit.net/jeux_peur/pacman-flash_4989.html

## III Qix

`Qix` is an arcade video game from 1981. The game present a space which contains a monster : the `Qix`. The player can move inside this space and leaves behind him a trail. When the player, after he left the border of the area come back to a border, the whole zone which was isolated from the `Qix` disappeir.

The player wins when the remaining space containing the `Qix` goes under 25% of the origin size.

The player loses if the `Qix` cross the trail.

Player's trail burn from its origin to the player itself if the player stop. If the fire catch him, he loses.

There is also monsters on the side of the space, including old player's trails. If one of these monster touch the player, he loses.

http://jayisgames.com/games/qix/
https://www.youtube.com/watch?v=nBt4w3qKI6I

## IV Centipede

`Centipede` is an arcade video game from 1981. The game features a game area with many empty space and some blocking box. The player is at the bottom of the screen and can move in every direction with some limitation : he can only go up and down a few. He can go to left and right without any limitation. The player can also shoot projectiles to the top of the screen.

Regularly, `Centipede` appeir on the top of the screen. They go from left to right or right to left and go down when they encounter an blocking box or screen borders. If one of them touch the player, he loses. A `Centipede` have a head, a tail and a body between these parts.

When a shot from the player touch a `Centipede`, the touched part transform into a blocking box and the body is splitted in two parts, becoming two different smaller `Centipede`. The `Centipede` which is from the tail part encounter the blocking box and go back on the next line.

Blocking box can be destroyed by several shots from the player if he insists a little.

There can be only one shot at once.

`http://my.ign.com/atari/centipede`

## V Solar Fox, Space Fighter

`Solar Fox` is an arcande video game from 1981. The game takes place in space and the player commands a space ship. The gaming zone is a grid. Some part of the grid contain a powerup that can be picked up with a shot, almost in the same way as `Pacman`.

The main difference with `Pacman` is the shape of the game zone, enemies and player's move.

- In `Pacman`, the game zone is a small labyrinth. In `Solar Fox`, the game zone is entirely available and without any obstacle.
- Ennemies in `Pacman` move inside the labyrinth while seeking the player. In `Solar Fox`, opponents are canon on the border of the game zone and they shoot in straight line.
- Last detail, in `Solar Fox`, the player cannot stop. He can shoot a small laser which can break powerups and intercepts ennemies shoots.

So, `Solar Fox` is a dodge and collect game. A lot of clone added functionnalities and a great speed that makes `Solar Fox` a typical arcade game.

https://www.youtube.com/watch?v=eTeC8Za9oSs
https://www.youtube.com/watch?v=ZKDgr2YxysA

# VI The project

`Arcade` is a software that allow the user to choose a game, to play and that keep score. When the program is launched, it musts display in separated box :

- All games available (In the directory ./games/)
- All graphic libraries available (In the directory ./lib/)
- Scores
- A field that allow the user to enter a name

## VI.1 Generalities

For your own culture it is very important to know several graphic libraries. This is why your final project must provides 3 different graphic libraries. The interest of this project being to make you handle dynamic libraries at the execution, the graphic rendering as well as output must be located within a dynamic library used for the execution. In order to optimize your software conception, you will also have to put your games inside dynamic libraries. The body of your program must interact in a uniform way with one or another of your libraries.

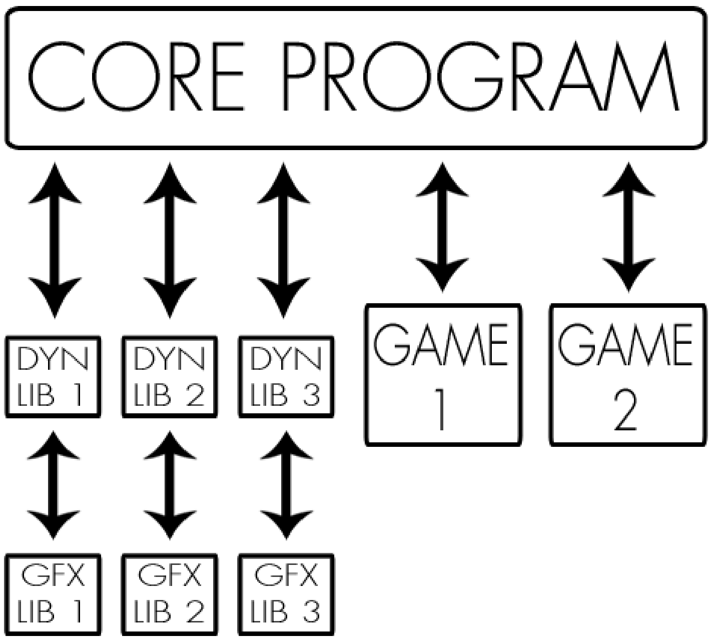

FIGURE 1 – `Architecture`

Each GUI available for the program must be used as a shared library that will be loaded and used dynamically by the main program. It is strictly **FORBIDDEN** to do any reference to one or another graphic library that you want to use in your main program. Only your dynamic libraries can do it ! Same thing for your games.

Again: It is strictly **FORBIDDEN** to do any reference to one or another graphic library that you want to use in your main program.

## VI.2 Dynamic libraries

You must use your dynamic libraries at the execution of your program. This means that you MUST use functions `dlopen`, `dlclose`, `dlsym` and `dlerror` to handle your dynamic libraries, as presented in class. Dependency to your libraries will not appears when passing your program as parameter to the command `ldd`.

Those libraries can be considered as plug-ins providing various capabilities for your main program. In NO CASE graphic libraries must influence the game logic. The same way, game libraries must not contain any information about screen rendering or low level events.

You MUST NOT make any difference between one or another library of your main program. Each of your libraries MUST be handled in a generic and uniform manner. It is the generic aspect that interests your grader during the project defense.

*Indices* Class and material available on the e-learning, as well as the tutorial are important sources of information to optimize the realization of this part of the project. However, nothing replaces the thinking.

### VI.3 Graphic libraries

You must choose three graphic libraries. Choose one per group.

Group 1 :

- NCurses
- NDK++
- aa-lib
- libcaca

Group 2 :

- Allegro4
- Allegro5
- Xlib
- GTK++
- SFML
- SDL1
- SDL2

Group 3 :

- OpenGL
- Qt
- MiniLibX
- LibLapin

If you want to use a graphic library that is not in this list, contact me to get an authorization. I recommend to contact me through the school social network on the intranet, so that your proposal can benefit your classmates too.

Some of these libraries cannot be installed on the standard dumps of the school. You should then consider that you will have to install manually these libraries, without privileged rights. You will acquire more than culture at this point.

## VI.4 Games : Nibbler

Rules are simple and must be respected.

- The unit measure is the "case". The size of a case is up to you, but it MUST be reasonable and CAN vary with the graphic libraries you are using. A 1px case is too small while a 1000px is too large.
- The game area is a finite plan of cases. The edges of the plan can't be passed through.
- The snake starts with a size of 4 cases in the middle of the game area.
- Le snake progress forward automatically at a constant speed. Each section of its tail follows the exact same path that the head.
- The snake can turn to the right or to the left from 90° when the corresponding touch of the keyboard is pressed.
- The goal of this game is to feed your snake so that he can grow. The game area MUST NEVER have less that one element of food.
- One food element fill only one case of the area.
- When the head of the snake is over a case with food, the food disappear, and one section of one case is added at the tail of the snake. The added section appears in the first free case nearby the last case of the tail of the snake. If there is no free case, the game is over. If a new section of the serpent is added, then a new food element appears.

Once you are done with games and the 3 graphic libraries, you can look for a bonus by extending the game rules. These are some examples :

- A bonus of food appears for a short period of time
- The head section looks different from the others sections
- The snake speed increases during the game
- The game area has some obstacles
- The size of the snake increases randomly when eating
- A speed boost limit when pressing the space bar
- ...

## VI.5 Games : Pacman

Rules are simple and must be respected.

- The unit measure is the "case". The size of a case is up to you, but it MUST be reasonable and CAN vary with the graphic libraries you are using. A 1px case is too small while a 1000px is too large.
- The game area have a specific size. A crossed sized make the player appeir on the opposed side. All case that are not wall are walkable and contain the “pacgum” powerup.
- In the middle of the map, there is a small area of 5 case width and 4 case height that contains ghosts.
- Ghost can get out of their box 10 seconds after the game started.
- Pacman starts the game right under the ghost.
- Some special, bigger “Pacgum” allow Pacman to eat ghosts. This effect is 10 seconds long. During this period, ghosts become blue and instead of hunting Pacman, they will try to escape. Their move speed is quite slow during this period. There is only 4 “Pacgum” of this kind available on each map.
- When Pacman eat a ghost, it only remains eyes. These eyes go back quickly to the ghost box where the ghost is fixed after a short period.
- The player win when Pacman ate all “Pacgum”. A new map is loaded right after, or the current one is reloaded and go faster.
- On screen, Pacman and ghost must not evolve case to case, but have a fluid movement.

Once you are done with your project and the 3 dynamic libraries, you can look for a bonus by extending the game rules. These are some examples :

- Food appeir a short time at the start position of Pacman. It provides powerup or huge score bonus.
- Pacman and a few ghost can jump, just like in `Pacmania` : https://www.youtube.com/watch?v=lX07p897vZU
- The game speed increase
- The game features a camera, like in `Pacmania`.
- ...

## VI.6 Games : Qix

Rules of the game are very basic and MUST be respected. These are the bases :

- The unit measure is the "case". The size of a case is up to you, but it MUST be reasonable and CAN vary with the graphic libraries you are using. A 1px case is too small while a 1000px is too large.
- The game area have a specific size. A case can contain three different values that indicate its type : the case can be walkable, it can be non walkable, it can be a border. A border is a specific walkable area.
- Non walkable case are space that were taken out by a player action. Borders are case that are directly in touch with a walkable case and at least one non walkable area.
- The walkable area contains a monster : the `Qix`, which measure several case and move in a random way (You have to fin a nice way to make him dreadful without having to hunt the player) If the `Qix` touch the player or his trail, the player lose and go back to the border he came.
- When the player go a a walkable zone, there is a trail behind him. This trail burn if he stops. When a touch a border after he walked inside the walkable area, the walkable area splits : the area which contains the `Qix` stays and the other become non walkable. Note that the player cannot cross its own trail.
- Borders contains specific monsters. Sparks can move on them. Sparks make the player loses by touching him. Sparks can also go on previous trail, old borders even if they are in non walkable zone, if it is interesting for hunting the player. A spark cannot turn back.
- To win, the `Qix` must be sealed inside less than 25% of the space.

Once you are done with your project and the 3 dynamic libraries, you can look for a bonus by extending the game rules. These are some examples :

- Powerups in walkable zone may appeir. The player may touch the powerup directly or grip it with its trail, depending your will. Type of powerups are on your imagination.
- Maps may contain obstacles or scenery.
- ...

## VI.7 Games : Centipede

Rules of the game are very basic and MUST be respected. These are the bases :

- The unit measure is the "case". The size of a case is up to you, but it MUST be reasonable and CAN vary with the graphic libraries you are using. A 1px case is too small while a 1000px is too large.
- The game area is splitted in two parts : the first is walkable and the other is not. The walkable area is at the bottom of the screen. The walkable part fill the whole width of the screen but only cover 20% of the height.
- From the top of the screen, centipede are coming. The go from left to right, or right to left and each obstacle or screen border make it change and go down to next line. A centipede is a snake constitued of several parts.
- The player can shoot. When a shoot touch a centipede, it is splitted in half. The touched part turn into an obstacle, the forward part goes on and the back part collide with the obstacle, turn back and progress to the next line.
- If a centipede touch the player, the player loses.
- If a centipede touch the bottom of the screen, the player loses score.
- To win, the player must survive 10 centipedes. The map is reseted and the game restart.
- A map for `Centipede` contains obstacles randomly generated.
- An obstacle can be destroyed by a 5 shot serie.
- There can only be one shot at once on the map.

Once you are done with your project and the 3 dynamic libraries, you can look for a bonus by extending the game rules. These are some examples :

- Several centipede at the same time and on the same map.
- Several type of centipede : length, speed, move.
- The player may represent two characters instead of one.
- . . .

## VI.8 Games : Solar Fox

Rules of the game are very basic and MUST be respected. These are the bases :

- The unit measure is the "case". The size of a case is up to you, but it MUST be reasonable and CAN vary with the graphic libraries you are using. A 1px case is too small while a 1000px is too large.
- The game area is splitted in two parts : a central part that is walkable and leave 2 or 3 case around, between its limit and the screen border. This is the walkwable zone. The other part is non walkable and contains opponents.
- The player evolves in the walkable zone. He can go in every direction, but cannot turn back directly. The player cannot break : he always go forward. There is even a key to go faster.
- The walkable area is filled with powerups that the player must break by shooting inside to win.
- Player opponents are close to the walkable zone and shoot.
- Opponents's shoots can destroyed by player's shoot.
- Player's shoot have only a two case scope. Its speed is three or four quickier than the player ship.
- The player loses if his spaceship is touched by a shoot. by special bad powerups or by hitting the walkable area borders.
- The shaceship, lasers, opponents, must not have a case per case movement. It must be fluid.

Once you are done with your project and the 3 dynamic libraries, you can look for a bonus by extending the game rules. These are some examples :

- Powerups on the walkable area.
- Opponents with different kinds of shoot or move.
- Obstacle, indestructible or not appeir in the game area.
- ...

## VI.9 Utilisation

The results of your turn in compilation MUST BE a program, 3 graphics dynamic libraries and at least two games. The program name MUST be "arcade" and the dynamic libraries MUST be named from the graphic or game library they are using. Only the executable name is imposed. However the libraries SHOULD have a name similar to "lib_arcade_XXX.so", where "XXX" is the name of the graphic library used or the name of the game.

The exec "arcade" must accept the graphic library to use.

Example :

```
>./arcade ./lib/lib_arcade_opengl.so
```

You MUST handle the following cases :

- If the number of argument passed to your program is different from 1, your program MUST display a usage and exit neatly.
- If the dynamic library does not exist or is not compatible, your program MUST display a relevant error message and exit neatly.

When your program is running, these keyboard touch MUST behave in the following way :

- '2' : Previou graphic library.
- '3' : Next graphic library.
- '4' : Previous game.
- '5' : Next game.
- '8' : Restart the game.
- '9' : Go back to menu.
- Echap : Exit.

# VII Mandatory Protocol

You MUST implement an IO system with STDIN and STDOUT in your game library. Your library must feature the function **void Play(void)** that start the game in this fashion. Our main will call this function. Your system MUST be synchronous, your game must not be played when no commands are sent. Here are the commands :

```
// Kind of Advanced Language Assistant Laboratory 2006-2042
// Epitech 1999-2042
// Jason Brillante brilla_a brilla_b
// Have you played Atari today?
//
// WELCOME TO THE ARCADE
// ENJOY OR DIE

#ifndef   __ARCADE_PROTOCOL_HPP__
# define  __ARCADE_PROTOCOL_HPP__
# include  <stdint.h>

namespace  arcade
{
  enum class  CommandType : uint16_t
  {
      WHERE_AM_I = 0,  // RETURN A WHERE AM I STRUCTURE
      GET_MAP  = 1,  // RETURN A GETMAP STRUCTURE
      GO_UP  = 2,  // MOVE THE CHARACTER UP
      GO_DOWN  = 3,  // MOVE THE CHARACTER DOWN
      GO_LEFT  = 4,  // MOVE THE CHARACTER LEFT
      GO_RIGHT  = 5,  // MOVE THE CHARACTER RIGHT
      GO_FORWARD = 6,  // MOVE THE CHARACTER FORWARD (FOR SNAKE)
      SHOOT  = 7,  // SHOOT (FOR SOLAR FOX AND CENTIPEDE)
      ILLEGAL  = 8,  // THE INSTRUCTION WAS ILLEGAL
      PLAY  = 9  // PLAY A ROUND
    };
  enum class  TileType : uint16_t
    {
      EMPTY  = 0,  // TILE WHERE THE CHARACTER CAN GO
      BLOCK  = 1,  // TILE WHERE THE CHARACTER CANNOT GO
      OBSTACLE  = 2,  // FOR CENTIPEDE
      EVIL_DUDE  = 3,  // EVIL DUDE
      EVIL_SHOOT = 4,  // EVIL SHOOT
      MY_SHOOT  = 5,
      POWERUP  = 6,  // POWERUP
      OTHER  = 7  // ANYTHING THAT WILL BE IGNORED BY THE KOALINETTE
    };
  /// The format is width, height, and width * height * sizeof(TileType) quantity of TileType
  struct  GetMap
  {
    CommandType  type;
    uint16_t  width;
    uint16_t  height;
    TileType  tile[0];
  } __attribute__((packed));
  /// The format is length, length * Position quantity of TileType
  struct  Position
  {
    uint16_t  x;
    uint16_t  y;
  } __attribute__((packed));
  struct  WhereAmI
  {
    CommandType  type;
    uint16_t  lenght;
    Position  position[0];
  } __attribute__((packed));
}

#endif //  __ARCADE_PROTOCOL_HPP__
```

## VIII Documents

More than any other kind of software, a software that can be extended like the Arcade project MUST have a documentation.

In order to clarify the way your program and interface work, you will write a documentation. Here are needed informations :

- You must provide a class diagram of your program, featuring at least link between classes and public functions.
- Some explaination in a manual that go with your diagram and describe how procedure are linked in your program.
- Also explain how to create a dynamic library for graphics and for gaming that is compatible with your system.

## IX Share your interface

We often have to collaborate with other teams. This collabe may occurs on the same project, but also sometimes on a shared formats.

In Arcade, there is three parts that can be shared : the score file format, the interface for graphic libraries and the interface for game libraries.

We ask you to collaborate with one other team. Establish a “standard” that contains description for your score file, graphic libraries interface and game libraries interface. You will have to both implement libraries and programs that use this “standard”.

`Pay attention to come together, at the same time, to your defense.`

## X Rules

You are more or less free to implement your program how you want it. However there are certain rules :

- The only functions of the `libc` that are authorized are those one that encapsulate system calls, and that don’t have a C++ equivalent.
- Each value passed by copy instead of reference or by pointer must be justified. Otherwise, you’ll loose points.
- Each value non `const` passed as parameter must be justified. Otherwise, you’ll loose points.
- Each member function or method that does not modify the current instance and which is not `const` must be justified. Otherwise, you’ll loose points.

- There are no `C++` norm. However if a code is reckoned to be unreadable or dirty, this code will be penalized. Be serious please!
- Keep an eye on this project instructions. It can change with time!
- I am very concern about the quality of my materials. Please, if you find out any spelling mistake, grammatical errors etc. please contact me at koala@epitech.net so that I can do a proper correction.

# XI Turn-in

You must turn in you project in the repository provided by `Epitech`. Repository name is `cpp_arcade`.

Repository will be cloned at the exact hour of the end of the project, intranet `Epitech` being the reference.

Only the code from your repository will be graded during the oral defense.

Good luck!