ESAT INNOVATION SCHOOL | ESCUELA SUPERIOR DE ARTE Y TECNOLOGÍA

ADVANCED PROGRAMMING
MEMORIA

Subject: Advanced Programming
Associated module: 2º HND in Computing
Teacher: Gustavo Aranda
Carrer time: 2020/2021
Authors: Álvaro Vázquez Carmona, Andres Carsana, Saul Camacho Diaz, Adam Alvarez

Índice/Index

1.- Explanation of an algorithm and an example of use Pág 03

2.- Explanation of programming paradigms and a resume into IDE use Pág 04

3.- Explanation of an IDE Pág 04

- 3.1.- Difference between develop with one or not Pág 04
- 3.2.- A little of explanation of how to use debugger Pág 05
- 3.3.- How a existence of a normative helps the companies Pág 07

4.- Database Pág 07

- 4.2.- Example of use into Database Pág 07
- 4.3.- Evaluation of the correct extraction into the database Pág 07
- 4.4.- Explanation of the use of the database into the project Pág 08
- 4.5.- Evaluation of the use of the database into the project Pág 10

5.- Guide of the team work and tasks Pág 12

6.- Short User Manual Pág 13

7.- Post-mortem Pág 15

1.- Explanation of an algorithm and an example of use.

An algorithm is a step sequence of instructions with the next shape:

- Code editor: the programmer tool for coding.
- Source Code: the source file of the programmer.
- Compiler: translates the source code to machine code.
- Object Code: the translation given by the compiler.
- Linker: establishes the link between libraries to finally make the executable file with the header of the OS system.
- EXE file: executable file.

In my case I used this algorithm to determine the position of the cursor into the tilemap, the first option is way more optimized than the other one, because at the frist option the Tile is being overwritten by the for bucle.

```
void Tilemap::set_cursorFirstMapPos(){

  Tile* tile = nullptr;
  bool found_free_tile_ = false;

  for(int i = 0 ; i < MKA::ListSize(grid_list_) && !found_free_tile_ ; ++i){

    tile = (Tile*)MKA::IndexListInfo(grid_list_, i);

    if(tile->walkable_){

      cursor_.init(tile->pos_);

      found_free_tile_ = true;

    }

  }
}
```

```
void Tilemap::set_cursorFirstMapPos(){

  bool found_free_tile_ = false;

  for(int i = 0 ; i < MKA::ListSize(grid_list_) && !found_free_tile_ ; ++i){

    Tile* tile = (Tile*)MKA::IndexListInfo(grid_list_, i);

    if(tile->walkable_){

      cursor_.init(tile->pos_);

      found_free_tile_ = true;

    }

  }

}
```

2.- Explanation of programming paradigms and a resume into IDE use.

- Procedural: includes an algorithm that has the job to generate something automatic, that there's no need to have a manual manipulation of. Algorithms Oriented Programming.

- Objects Oriented Programming: programming oriented to data.

- Events Oriented Programming: asynchronous interruptions to the system (callbacks).

The most used programming tool to give the programmer all the possibilities to use any of this paradigm is Visual Studio with his powerful debugging tools and the facilities that it incorporates.

One of this programming paradigms that I used in my project is the Objects Oriented with the next structure at the different C++ files.

- Include.h
    - Define symbol
    - Includes
    - Forward declarations
    - Class declaration
    - public methods or variables
    - protected methods or variables
    - private methods or variables

- Source.cc
    - Includes + Class include
    - Static variables
    - Class methods

3.- Explanation of an IDE.

It's an environment where the own programmer has the necessary tools to his development.

3.1.- Difference between developing with one or not.

It depends on the programmer, for example the visual appearance is not attractive for everyone. But one disadvantage of working without one is that you have to work with separate tools, instead an IDE has all that tools integrated in one application.

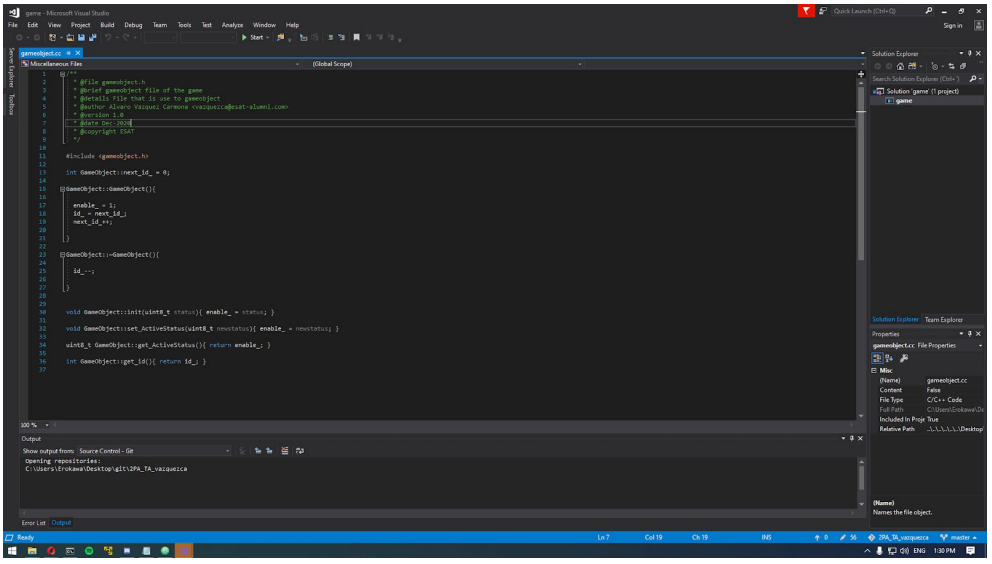
Taking advantage of using an IDE can be long term bad, because if you are going to work in other site with a different environment It can be painful.

3.2.- A little explanation of how to use debuggers.

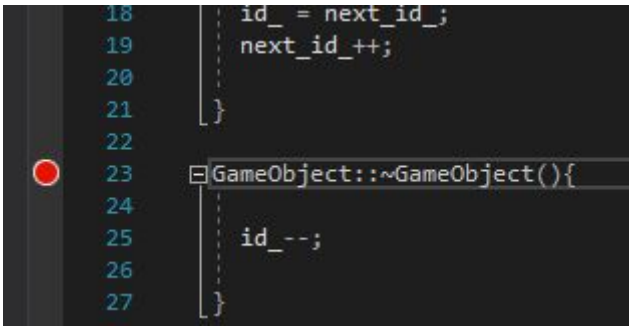
The debugger it's a very useful tool to find errors because the program crashes or to watch the values into the variables that the programmer is using.

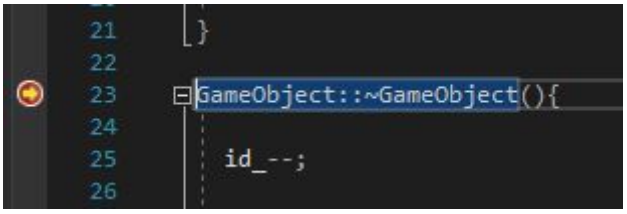
The next command into the developer command prompt to debug an application with source files.

```
\2PA_TA_vazquezca\esat_rpg\build>devenv game.exe
```

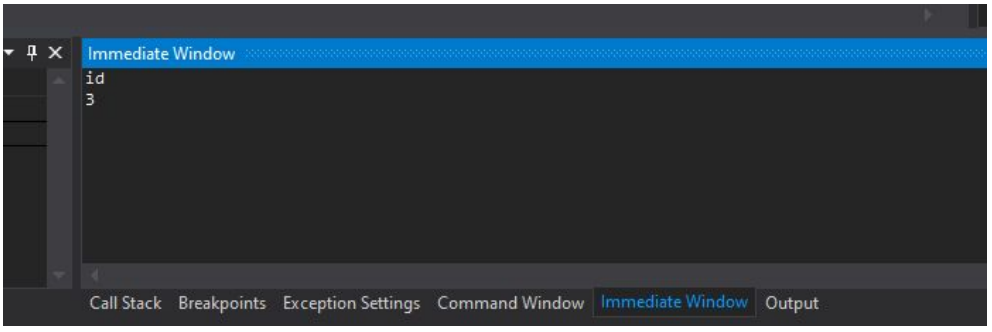


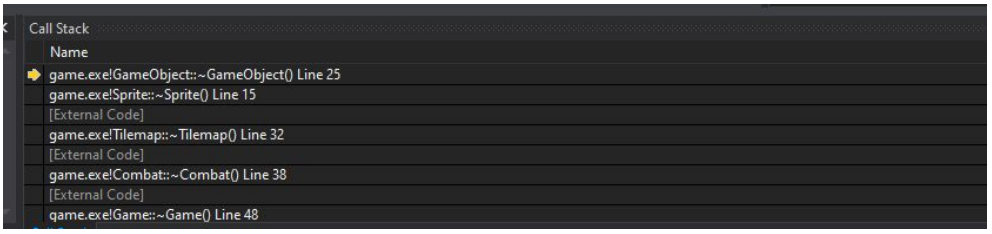
The breakpoint it's a interruption point where you can follow the program instructions step by step. You can use F10 to go by instructions step by step, F11 if you want to go inside each instruction.





The immediate Window tells you the values of a variable or if it’s a pointer the memory direction and the content is pointed by that are inside the function stack.



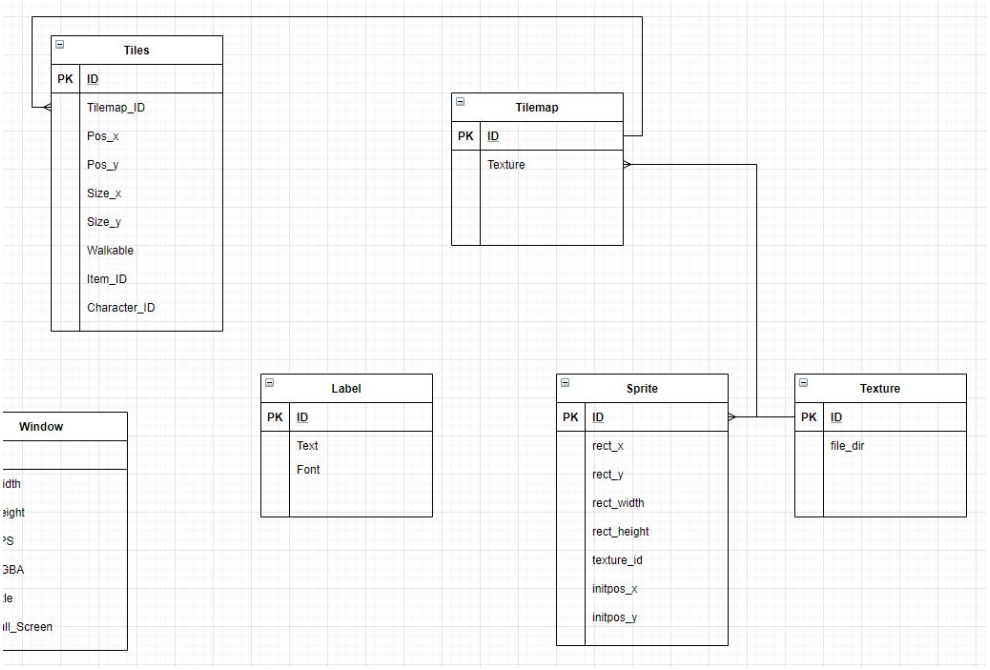


3.3.- How the existence of a normative helps the companies.

The normative into a corporation garanties a correct organization of the code, this means that anyone has to be able to understand the code you are writing. Besides, if someone of the company moves to another part of the company and there comes a new employee that touches the same code was writing the last one, it makes it even more easy and improve the own workflow of the office.

4.- Database.

The table made for the project has been made with an draw tool:



4.2.- Example of use into Database.

For this project the database needed to include the data game, for example the paths of the textures used into the game or maybe the characters' stats. That makes the data insertion into a video game so easy to do essentially because the programmer doesn't need to recompile the code, because the code already takes what is into the database.

4.3.- Evaluation of the correct extraction into the database.

The correct extraction of the data with a query can be used after into the code, for example you can have an UI with descriptions, that descriptions won't be hardcoded into the source code. Instead of that code, It can be a string that it takes the code from database.

4.4.- Evaluation of the use of the database into the project.

At the project I developed a Database class that makes the use of the queries even more interesting, because you don't need to code table queries and recompile all times you need. With this algorithm reads the content of an plain text file and it applies directly into the program giving it to the sqlite3 API.

```
uint8_t Database::sqlTableQuerie(const char* sql_c){

  open();

  int rc = sqlite3_exec(db_, sql_c, 0, 0, &errmsg_);

  if (rc != SQLITE_OK) {

    printf("\nSQL error: %s\n", errmsg_);

    sqlite3_free(errmsg_);
    sqlite3_close(db_);

  }

  close();
```

```
527 void Database::applyTableQueries(const char* table_q_f){
528
529   FILE *temp_file = fopen(table_q_f, "r");
530
531   if(nullptr != temp_file){
532
533     char character;
534     std::string buffer;
535
536     while(fread(&character, sizeof(character), 1, temp_file)){
537
538       if(character != '"' && character != ';'){
539
540         buffer += character;
541
542       } else if (character == ';'){
543
544         sqlTableQuerie(buffer.c_str());
545         buffer.erase(buffer.begin(), buffer.end());
546
547       }
548
549     }
550
551     fclose(temp_file);
552
553   }
554
555
556
557 }
```

4.5.- Explanation of how the database works.

The database works with queries, a queries its the SQL syntax that allows you to take the data you want from the database. For example:

“SELECT * FROM CHARACTERS”
This query tells to the database that wants to select all the data of the table characters,

“INSERT INTO CHARACTERS VALUES(HP)”
This query tells to the database that wants to insert a new row with a determined value,

In the code, at the project I created specific functions that calls API’s functions to open and close the database each time a query is done.

```
void Database::open(){

    int rc = sqlite3_open(path_, &db_);

    if(rc != SQLITE_OK){

      printf("\nSQL error: %s\n", sqlite3_errmsg(db_));
      sqlite3_close(db_);

    }

}

void Database::close(){

  if(nullptr != db_){

    sqlite3_close(db_);

  }

}
```

The function that does the queries always opens and closes itself, this makes the code more efficient because the database opens when is needed.

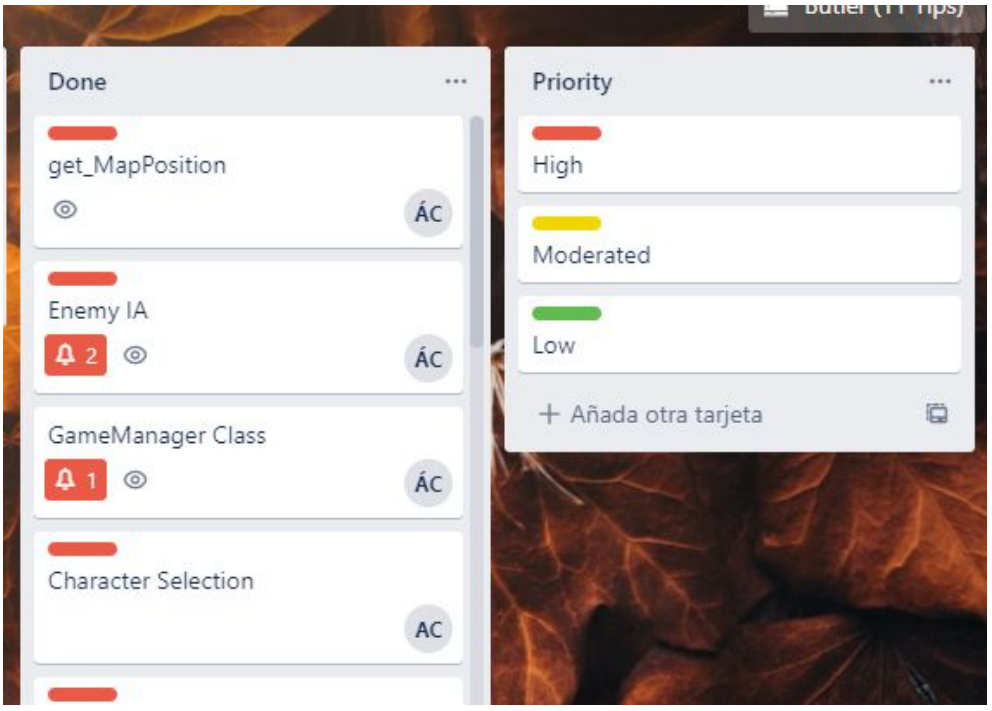
```
uint8_t Database::sqlTableQuerie(const char* sql_c){

  open();

  int rc = sqlite3_exec(db_, sql_c, 0, 0, &errmsg_);

  if (rc != SQLITE_OK) {

    printf("\nSQL error: %s\n", errmsg_);

    sqlite3_free(errmsg_);
    sqlite3_close(db_);

  }

  close();
```

5.- Guide of the team work and tasks.

The developer team has used Trello to organize the different tasks. Each of us chose a different task to develop.

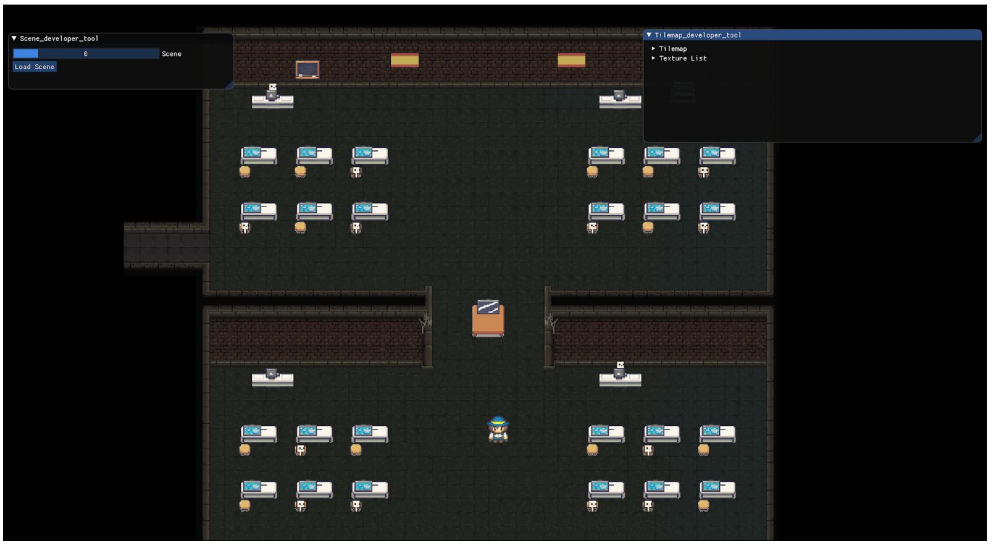
The task that we had into this project we have differentiated them by priority with colors. That helped us to determine which tasks has a higher development priority and follow a guide to do the game in the right direction.

Some of the tasks had time to be developed, for example Enemy IA has to be developed as much as possible in 1 week.

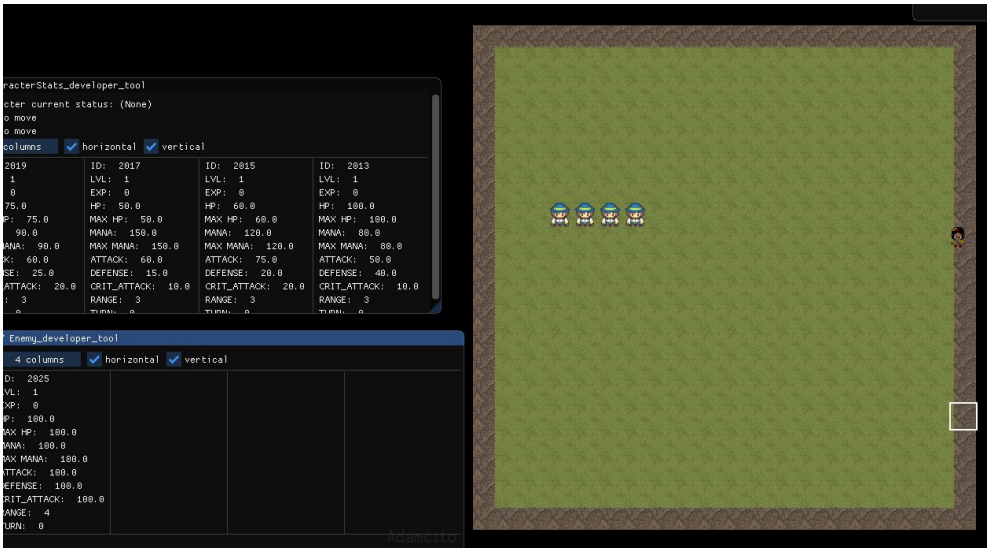


6.- Short user manual.

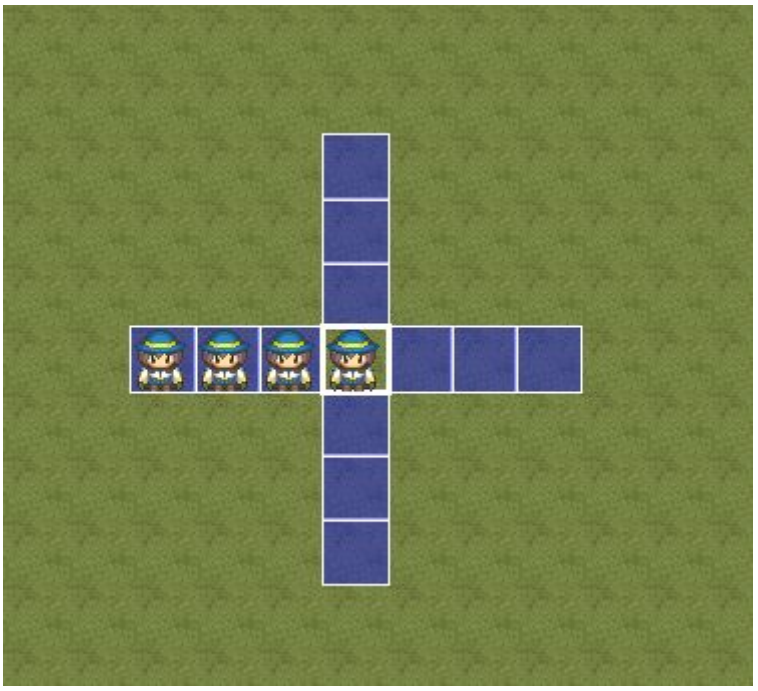
The game starts on an open world map, where the player can move anywhere.



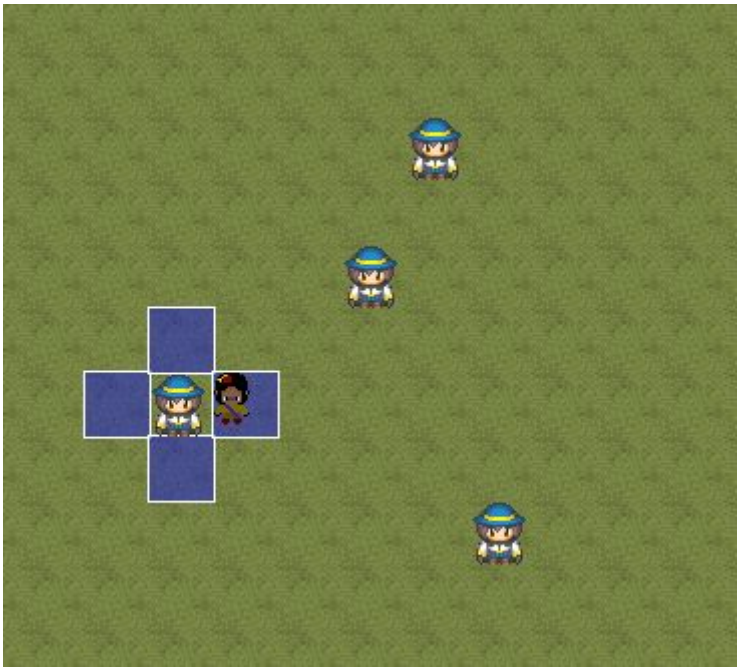
At some tiles the player will find confrontations against enemies, and that enemies will target a random player finding the closer path to go.



Selecting a character will make appear the range of the character to move or attack.



When the turns of the characters are done, the enemies will make the closer path to the target.



7.- Post-mortem.

We learned a lot doing this project about C++ and in my case IA algorithms and the entire tilemap and combat. We didn't finish the game, work was missing on some team components and sometimes deadlines for work were not met.