ESAT INNOVATION SCHOOL | ESCUELA SUPERIOR DE ARTE Y TECNOLOGÍA

Mathematical library with procedural animations

Nombre Asignatura: Prog. Avanzada
Módulo Asociado: 2 VGP MA
Profesor: Gustavo Aranda
Curso: 2023/2024
Autor/es: Carlos Mazcuñán Blanes & Lucas Calatayud Briva

Índice/Index

1.- Algorithms

An algorithm is a set of instructions that a computer must follow to solve a problem or perform a task. An algorithm can be expressed in different ways, such as natural language, pseudocode, flowcharts, or programming languages

Algorithms are used for a variety of reasons:

- **Efficiency**: They are capable of executing tasks with speed and precision, making them indispensable for tasks that involve extensive calculations or data processing.
- **Consistency**: Algorithms can be run repeatedly and they yield uniform results each time, which is crucial when handling vast data or intricate processes.
- **Scalability**: They have the ability to be expanded to manage large datasets or intricate problems, making them beneficial for applications that necessitate processing substantial amounts of data.
- **Automation**: Algorithms have the capacity to automate monotonous tasks, lessening the need for human involvement and allowing more time for other tasks.
- **Standardization**: Algorithms can be standardized and disseminated among various teams or organizations, facilitating collaboration and knowledge sharing.

Implementation of the algorithm (state machine)

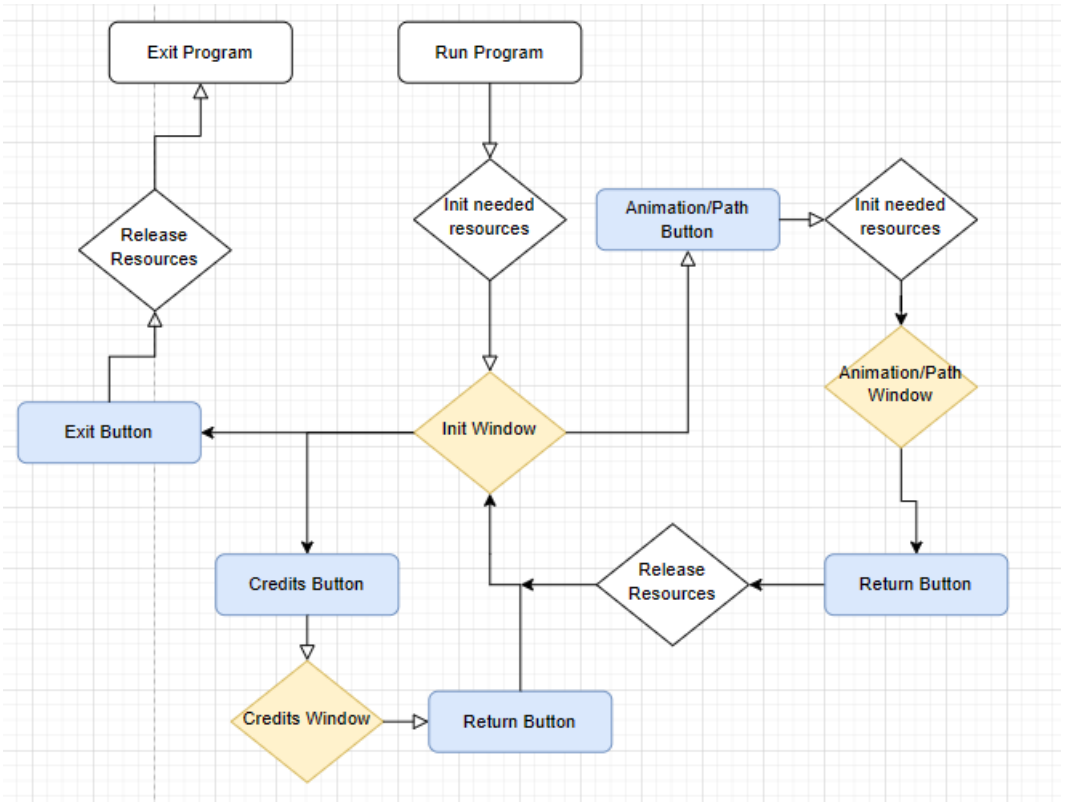

Fig 01

Once the algorithm has been created and it is clear what we want to do, to continue creating the application, we begin to write the source code of the program in the editor (such as visual studio code) that will contain the instructions of the program. , then, once all the source code is written, it is compiled using a compiler such as in our case Visual Studio to generate the object code, then a linker is used to combine this object code with the necessary libraries to create an executable.
Finally, the executable program is loaded into memory and executed.

2.- Programming Paradigms

Programming paradigms are ways of classifying programming languages based on their characteristics. Some of them are procedural programming, object-oriented programming and event oriented.

Procedural programming consists of dividing the code into logical sections called routines or procedures, where each procedure solves a specific task, and is executed when necessary. It is applicable in both low-level and high-level programming languages.

```
void updateAnimationVolbeat()
{
  GameManager &GM = GameManager::Instance();
  for(int i = 0; i < 2; i++){
    if(GM.all_sprites.volbeat[i].animation_->config_.is_moving == 0)
    {
      GM.all_sprites.volbeat[i].phase_++;

      if(GM.all_sprites.volbeat[i].phase_ > 17)
      {
        GM.all_sprites.volbeat[i].phase_ = 0;
      }

      switch (GM.all_sprites.volbeat[i].phase_)
      {
      case 0:
        GM.all_config_.ac_volbeat_ = {
                  1, 0, 0,
                  {GM.windowSize.x / 1.5f, 225.0f},{GM.windowSize.x / 1.5f / 0.5f, 225.0f},2.5f,
                  0.0f,0.0f,0.0f,
                  {1.0f, 1.0f},{1.0f,1.0f}, 1.0f};
```

Fig 02

Some of the advantages of procedural programming are:

- **Optimization of code efficiency**: The segmentation of the program into more compact and optimized procedures allows to achieve greater speed and performance during the execution of the code. This also avoids code redundancy and promotes its reuse.

- **Maintenance and debugging simplified**: A modular and well-structured program facilitates the identification and correction of errors, as well as the implementation of changes in the code. Moreover, it improves the readability and comprehension of the code, simplifying long-term maintenance.

- **Flexible and versatile algorithm design**: The ability to create procedures to address problems or perform tasks allows the program to adapt to various situations or requirements. Moreover, different procedures can be combined to develop more complex and functional programs.

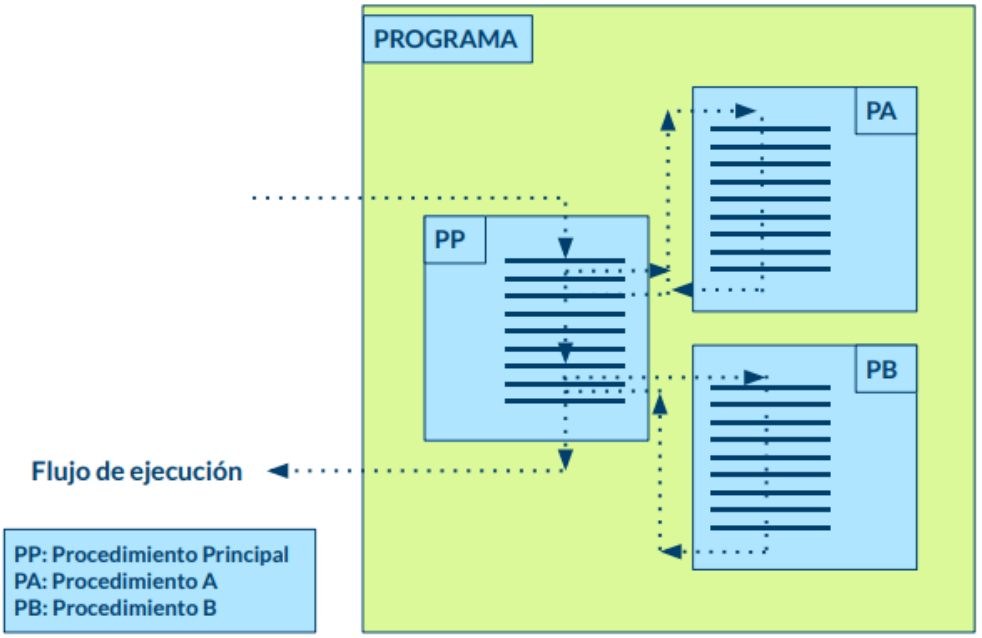

Fig 03

Object-Oriented Programming is a specific approach to programming where the code is structured into entities known as classes. These classes are used to create objects that interact with each other to achieve the purposes of the applications. An object serves as an abstraction of an entity, encapsulating data through properties or attributes and actions through methods.
Some of the advantages of this programming paradigm include:

- **Reusable Code**: The classes and objects created in a program can be reused in other programs. This can save time and effort in software development.
- **Reduction of Redundancy**: By reusing code, unnecessary duplication of code is avoided. This makes the code more efficient and easier to understand.
- **Easy to Maintain**: Object-oriented programs are usually easier to maintain and modify than procedural programs. This is because changes in one part of the code do not necessarily affect other parts.
- **Security**: Object-oriented programming allows restricting access to certain methods, properties, and classes through access modifiers like private and protected. This can help prevent errors and protect data

Fig 04

Event-driven programming is a type of programming where the structure and execution of programs are guided by events. These events can be user-defined, self-generated, or triggered by the system itself.

The advantages of the event-driven programming are:

- **Faster Processing**: Due to the propagation of data among multiple processors and event handlers, faster processing is achieved.

- **Interactivity**: This is the most significant advantage of event-driven programming, enhancing the user experience. Modern users want to actively participate and not remain as passive consumers. Therefore, event-based programs help users navigate their own experience and explore a variety of options, without pre-established restrictions.

- **Less Decision Coding**: This programming approach has numerous advantages for all stakeholders, as fewer decisions need to be coded than traditional applications. Many events are determined by user actions, rather than having advanced coding for all possible scenarios. While traditional applications act, event-driven applications react.

- **Flexibility**: It perfectly adapts to recent software engineering trends, such as cloud computing, microservices, and the need for better scalability with flexible and distributed systems.

- **Valid Inputs**: It eliminates the possibility of inputs with invalid values. A traditional application asks a question and allows the user to write a response. Event-driven applications usually offer a set of options, either as buttons or as drop-down menus.

- **Easy Maintenance**: The need to correct and adjust existing code when adding or removing some module is eliminated. The system continues to operate smoothly, regardless of any such adjustment.

```
if (ImGui::Button("Animation", ImVec2(100.0f, 20.0f)))
{
  GM.music_counter_ = 0.0f;

  initAllEntityParallax();
  initAllEntityCharacter();
  initAllAnimationConfig();
  windowManager(2);

}
```

Fig 05

3.-Implementation and Debugging

To implement the code of our program we have used visual studio code, since it is the code editor that we have used the most and to which we are most accustomed. We have also chosen vscode for its intelligent autocompletion or its help in detecting syntax errors in the code. And for the debug we have used the visual studio debugger.

The advantages of using this IDE instead of not having one are the previously mentioned advantages (intelligent autocompletion, help in detecting syntax errors in code…) Also thanks to its different plugins, we get very useful tools to work with, such as liveshare so that both partners can work locally even if we are not physically together, extensions to facilitate writing code in almost any language, git extensions to facilitate management of the repository, in addition to the fact that visual studio code is a very customizable tool so each member of the team can have a more comfortable workspace for themselves.

A debugger or debugging tool is a computer program used to test and debug other programs (the "target" program).The primary function of a debugger is to execute the target program in a controlled environment, enabling the developer to follow its operation and observe alterations in computer resources that could reveal defective code.

The IDE used to debug has been visual studio and we have used it to fix program errors such as memory leaks, runtime crashes…

When there was a problem with the program, we opened visual studio and opened it in its controlled environment, once we did that, if we couldn't figure out where the error was with the naked eye, depending on what the problem consisted of, we would check the call stack, inspect the variables in the program flow and we use breakpoints until we find the error and finally solve it.

We have tried to follow the ESAT programming regulations at all times when developing our application. The existence of programming regulations is useful for a company in cases like the code is passed from one employee to another, the time to get used to the code is reduced since it is easier to identify the parts of it when that code is in the hands of someone new who hasn't seen it before.

It must be emphasized that the programming regulations must be used as long as the engine used does not already have its own regulations, such as unreal engine, which does have its own regulations.

4.- Personal Tasks and Workgroup Plan

Although we have both been working on all the program's source code files, we have shared responsibility for them as follows:

Lucas: In my case I have taken responsibility for the files *animation, game_manager, entity* and *path*. I have also looked for the sprites used for the work

Carlos: In my case I have taken responsibility for the files *app_window, game, gsprite* and *texture*. Also I have looked for the samples and set the color themes.

5.- Short User Manual

- **Welcome window**

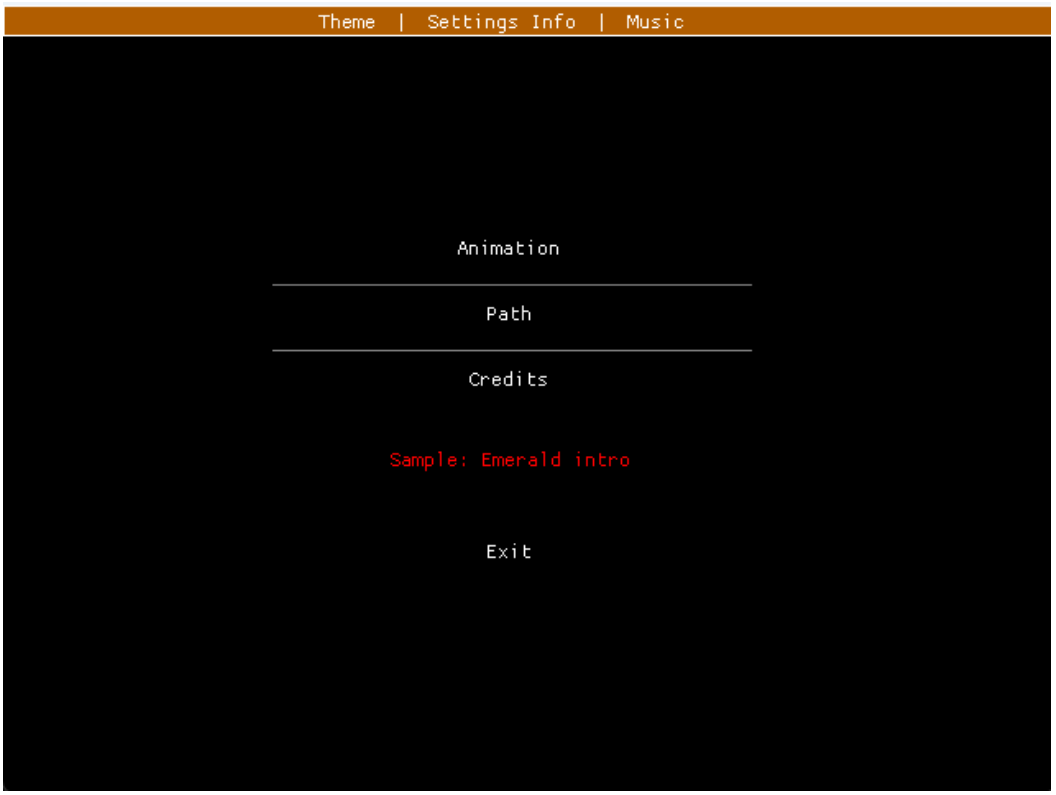

Fig 06

This window is the one that appears as soon as you run the program. In the center of it we find 3 main buttons: *Animation*, to access a demo of sprite animations based on the intro of Pokemon Emerald; *Path*, to access a demo in which we can modify a polygon; *Credits* which will open a window with information about the authors. At the bottom, the *Exit* button to exit the program.

At the top we find a menu bar with several functionalities:

Fig 07

The first. *Theme*, to change the theme or colors of the program:

Fig 08

The second, *Settings Info*, shows us the information about the resolution and fps, and ub the third option we can select which song we want to play when we access the *Animation* window. There are 4 to choose from, and the chosen sample will appear on the screen.

Fig 09

Fig 10

- **Animation**

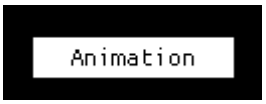

Fig 11

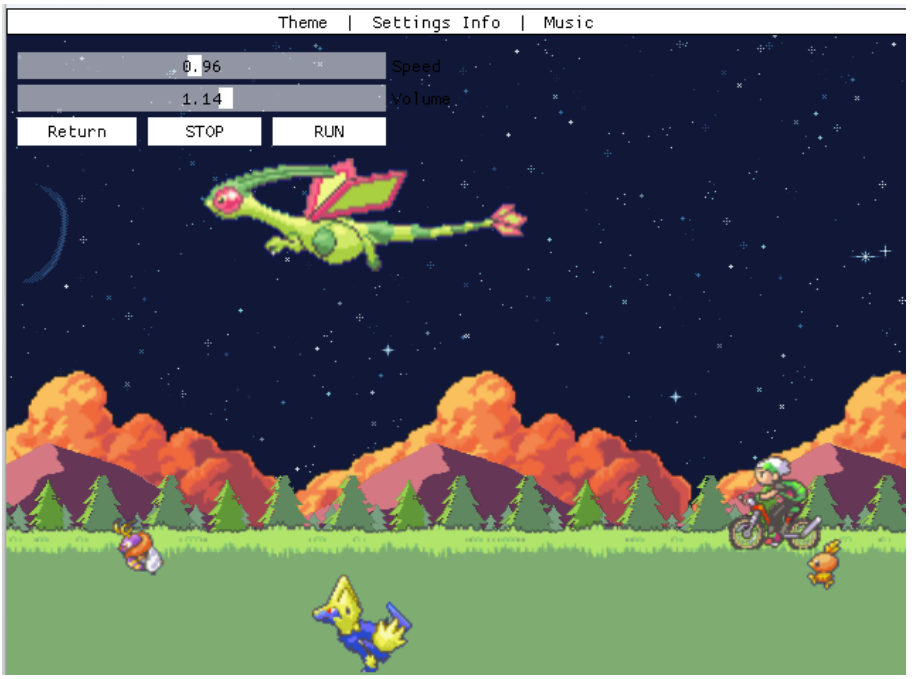

Fig 12

In this window we can see a small demonstration of how we use the Sprite class together with the Animation class. In which the sprites are moved with a predetermined movement, and to a lesser extent, some are scaled and rotated.

In the background, a *parallax* effect is achieved by drawing several layers on the screen and making each of these layers move at different speeds, with the furthest layer being the slowest (the night sky) and the closest being the fastest (the grass).

At the top left we have a small window with several functionalities:

Fig 13

The first bar is for modifying the speed of *movement*, the second for increasing or decreasing the *volume* of the audio. Below these there are 3 buttons, from left to right, the first, *Return*, to return to the beginning, the next, in the middle, *STOP*, so that the speed becomes x0 and the movement stops. and the last one, *RUN*, so that the movement speed becomes x1.

- **Path**

Fig 14

Fig 15

In this part, the Path class is used. A triangle appears under a window. In this window we can modify several polygon variables, starting with the *transform: position (x,y), scale(x,y), rotation*. Further down, adding or removing vertices from the polygon, the polygon cannot have less than 3 vertices.
Below are certain parameters to change the color of the border and fill, in *rgba* format and then a checkbox to select if you want the polygon to be solid or hollow.
Here's an example:

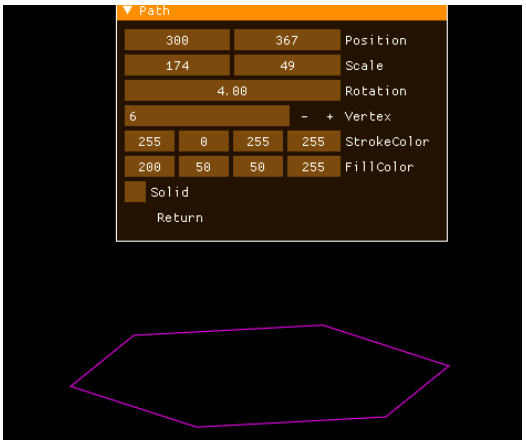

Fig 16

6.- *Conclusions and Future Work*

We have really enjoyed doing this work since with the freedom that has been left to us to focus the work we have been able to do it in a way that we have enjoyed by choosing to make an animation of a game saga that we like such as Pokemon, we have faced new challenges doing this work such as including new libraries that we had not used before such as Solaud or facing object-oriented programming which we had not applied to a project similar to this one. Despite having had problems during the development of the project, we consider that we have solved them satisfactorily and we have been able to learn a lot from this work.

As for things to improve in our program, we would have liked to not have certain parts so hard coded but with the time available we have not been able to think of a way to fix this. Another point of the program that could be improved is that it is limited to a fixed resolution since originally we had planned to apply different resolutions for the program window but with the problems that arose when creating the project we were not able to finally implement it.

7.- BIBLIOGRAPHY

GEEKSFORGEEKS, 2023e. Definition types complexity and examples of algorithm. GeeksforGeeks. Online. 16 October 2023. Available from: https://www.geeksforgeeks.org/what-is-an-algorithm-definition-types-complexity-examples/

VEGA, Héctor, 2023. Programación procedural. /Clases/2211-historia-programacion/35075-programacion-procedural/. Online. 8 August 2023. Available from: https://platzi.com/clases/2211-historia-programacion/35075-programacion-procedural/

JIMÉNEZ, Leonel and JIMÉNEZ, Leonel, 2023. Eficiencia y versatilidad del lenguaje procedural en el software. Online. 12 July 2023. Available from: https://leojimzdev.com/eficiencia-y-versatilidad-del-lenguaje-procedural-en-el-software/#desventajas_del_lenguaje_procedural

COLABORADORES DE WIKIPEDIA, 2023b. Programación por procedimientos. Wikipedia, La Enciclopedia Libre. Online. 7 September 2023. Available from: https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_procedimientos

Qué es la programacion orientada a objetos, [no date]. DesarrolloWeb.com. Online. Available from: https://desarrolloweb.com/articulos/499.phpVega, H. (2023, August 8). Programación orientada a objetos. Availablefrom:/Clases/2211-historia-programacion/35076-programacion-orientada-a-objetos/. https://platzi.com/clases/2211-historia-programacion/35076-programacion-orientada-a-objetos/

CORVO, Helmut Sy, 2020b. Programación orientada a eventos: características, ejemplos, ventajas, aplicaciones. Lifeder. Online. 30 March 2020. Available from: https://www.lifeder.com/programacion-orientada-a-eventos/

JOSE and JOSE, 2023. La programación desde su origen griego ha evolucionado hasta nuestros días.… Qué Es. Online. 2 October 2023. Available from: https://quees.com/programacion-orientada-eventos/

WIKIPEDIA CONTRIBUTORS, 2023. Debugger. Wikipedia. Online. 4 October 2023. Available from: https://en.wikipedia.org/wiki/Debugger

El proceso de compilación, del código fuente al código máquina, [no date]. Available from: https://uhu.es/javier.fernandez/Tema3.pdf

Figures:

Fig 01 - Screenshot of draw.io (diagrams.net)

Fig 02 - Screenshot of the application code in visual studio code.

Fig 03 - https://platzi.com/clases/2211-historia-programacion/35075-programacion-procedural/

Fig 04 and 05 - Screenshot of the application code in visual studio code.

Fig 06 to 16 - Screenshot of the program execution