ESAT INNOVATION SCHOOL | ESCUELA SUPERIOR DE ARTE Y TECNOLOGÍA

Advance Programing

Math Project

Subject: Advance Programing
Associated Module: 2º PA

Teacher: Gustavo Aranda
Author: Marcos Jiménez Saz and Javier guinot Almenar

Index

1º Algorithms

2º Programming paradigms
- 2.1º Procedural
- 2.2º Objects
- 2.3º Events
- 2.4º Comparison

3º Implementation & Debugging

4º Personal tasks & Workgroup plan

5º Short User Manual

6º Conclusions & Future work

1º Algorithms

- An algorithm is a sequence of unambiguous instructions that allow us to solve a problem or perform a complex calculation, in addition to other activities.

```
int i = 0;
for (int row = 0; row <= res_; row++)
{
  for (int column = 0; column < (2 * res_); column++)
  {
    (points_ + i)->z = ((float)sin(row * increment)) * ((float)cos(column * increment));
    (points_ + i)->x = ((float)sin(row * increment)) * ((float)sin(column * increment));
    (points_ + i)->y = ((float)cos(row * increment));
    i++;
  }
}
```

- This algorithm obtains points based on a sphere according to the amount of resolution of the sphere, resolution being a grid of rows x columns where all vertices are located.

- To improve the algorithm for obtaining the sphere we used a <= in lin2 29 of the file mathProject > src > sphere_3d.cc instead of a < which is the original algorithm. Larger reservation of the number of points is also made.

2º Programming Paradigms

2.1º Procedural

- It is about function-based programming and structured code.

2.2º Objets

- It is about programming based on classes which define their own attributes and methods (Functions internal to the class only accessible by an instance of said class or inheritance. With the possibility of being static, accessible by others)

```
class Sphere : public Entity
{
private:
  // Gets the points for Entity.
  void obtenerSphere();

public:
  Sphere(){};

  int init(SDL_Color color, bool fill = false,

  ~Sphere(){};
};
```

- Here we have the sphere class that inherits from entity. This class does not contain its own attributes but it does contain methods which are used to make an entity instance of sphere type and an empty destructor since the entity attribute destructor is in that class. This capture is only the class declarations but in the file src/sphere.cc you have the declarations.

2.3º Events

```
void Render::inputs(Keys *keys)
{

  if (EVENT_DOWN(UP, keys))
    rotation(right_);
  if (EVENT_DOWN(DOWN, keys))
    rotation(left_);

  if (EVENT_DOWN(RIGHT, keys))
    rotation(up_);
  if (EVENT_DOWN(LEFT, keys))
    rotation(down_);

  if (EVENT_DOWN(K_n, keys))
    rotation(front_);
  if (EVENT_DOWN(K_m, keys))
    rotation(back_);

  if (EVENT_DOWN(K_w, keys))
    translation(back_);
  if (EVENT_DOWN(K_s, keys))
    translation(front_);

  if (EVENT_DOWN(K_a, keys))
    translation(right_);
  if (EVENT_DOWN(K_d, keys))
    translation(left_);

  if (EVENT_DOWN(K_q, keys))
    translation(down_);
  if (EVENT_DOWN(K_e, keys))
    translation(up_);
}
```

- It is programming based on events caused by the user or the system.

- This function takes the corresponding inputs to move or rotate the camera around the program world.

2.4º Comparison

- The object oriented programming is used for the programmers to create the game basics & the event programming is used to modify/control the game.

3º Implementation & Debugging

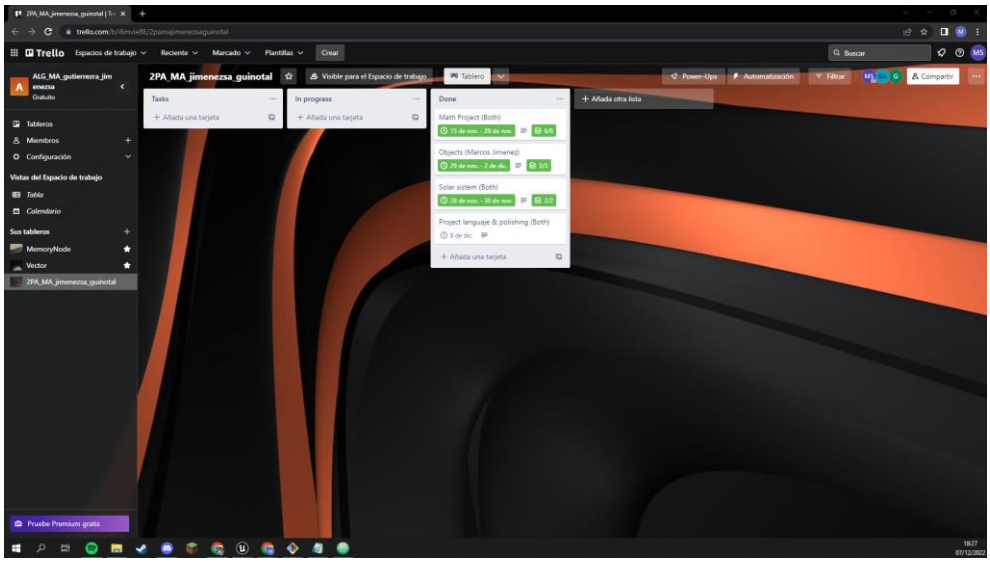
- To develop the application we have used Visual Studio community and/or Visual Studio Code in order to find memory errors, load the objects, check the values as they correspond and above all the 3d rendering of all the vertices of each object.

- In order to run a compiled application in debug from visual studio we must execute the command “devenv file.exe file.cc”. Once this is done, it will open a visual studio window with the .cc in which we will be able to set breakpoints, check variable values, call stack, memory, etc.

- To do this from visual code we need to create a folder called .vscode in which the .json that will be executed when compiling and/or executing the program in question will be stored. When we open the project folder that contains the .vscode folder, the VSCode interface will open and we simply compile with ctrl + shift + b and execute with F5 to be able to debug.

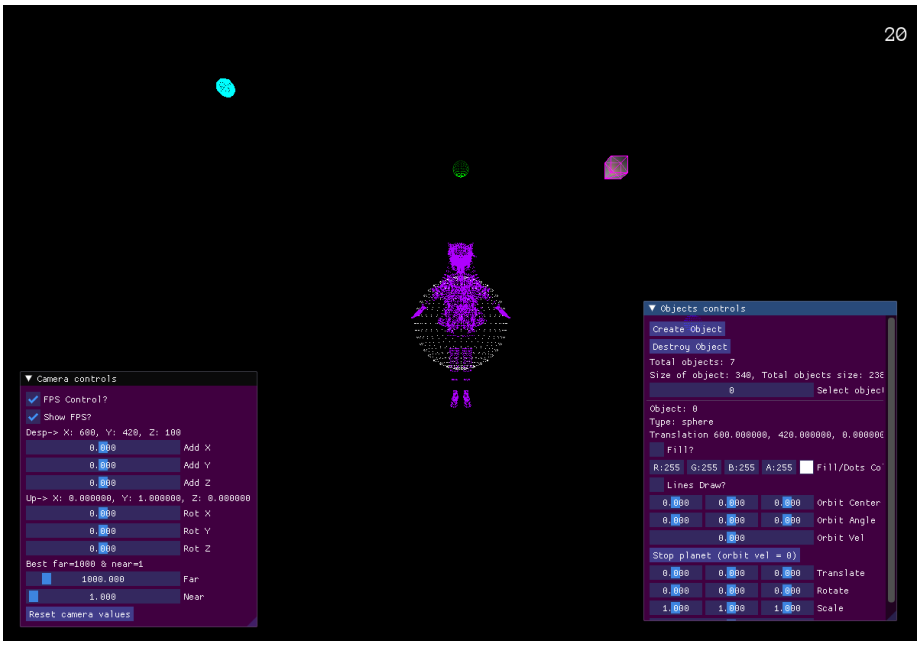
- It is possible to make a game without the need to use an IDE, but it would be much more complex to keep an eye on the algorithms of the program and the memory used by it. That is why it is advisable, even if only for debugging, to make use of an IDE.

4º Personal tasks & Workgroup Plan



https://trello.com/invite/b/i6mvieBE/ATTI5a930e9daabeba3284553380a1b31483B1BDB3FC/2pamajimenezsaguinotal

5º Short User Manual



The first look when you open the application. We can see the program and 2 windows to control the camera and objects.

Inputs:

- We can use the keyboard to move around this fictional world.

- ‘q’ & ‘e’: These keys move up & down the camera.

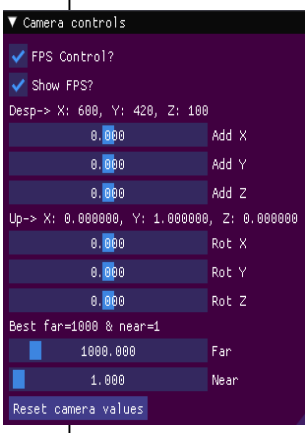
- ‘w’ & ‘s’: These keys move forward & backward the camera

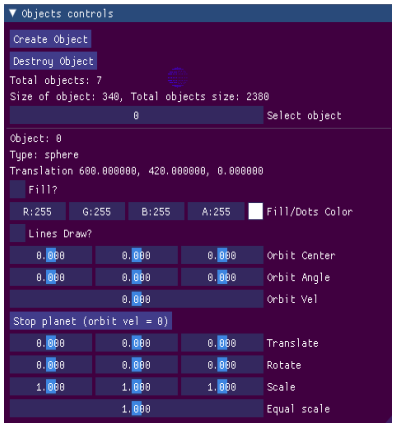
- ‘a’ & ‘d’: These keys move left & right the camera

- ‘m’ & ‘n’: These keys rotates in local z axis the camera

- ‘left arrow’ & ‘right’: These keys rotates in local ‘y’ axis the camera

- ‘up’ & ‘down’: These keys rotates in local ‘x’ axis the camera

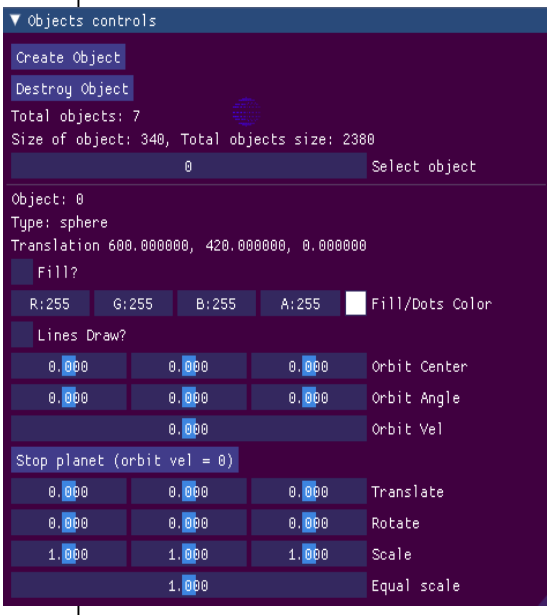




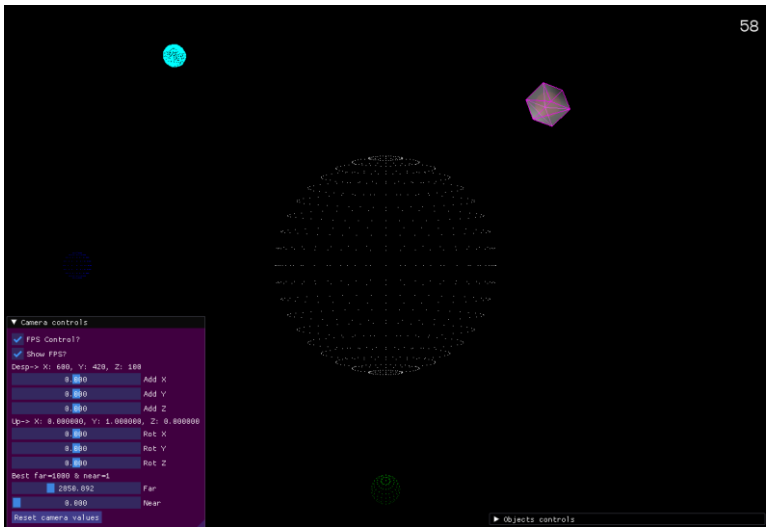
These are the 2 windows that appear when you open the application with the default values.

- With the window on the left you can move the camera around the screen.

Fields:



- **FPS Control:** With this field, you can set the maximum fps at which the application can run, which in this case will be 58, otherwise the maximum fps will depend on where the application is opened.
- **Show FPS:** As the name of the field indicates, this will be used to see or not the fps at which the application is running.
- **Desp:** These values are used to move the camera in the x, y, and z axis.
- **Up:** These values are used to rotate the camera on the x, y, and z axis.
- **Best:** These values are used to change the rendering distance. If these data are to be changed, this must be done before moving or rotating the camera on any of its axes

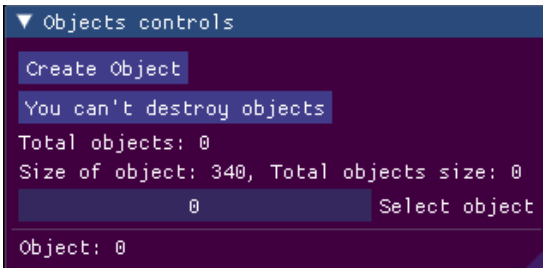


- **Reset camera values:** As the name suggests, it resets all camera settings to their default values.

- With the window on the right you can change the values of all the objects that can be seen on the screen.

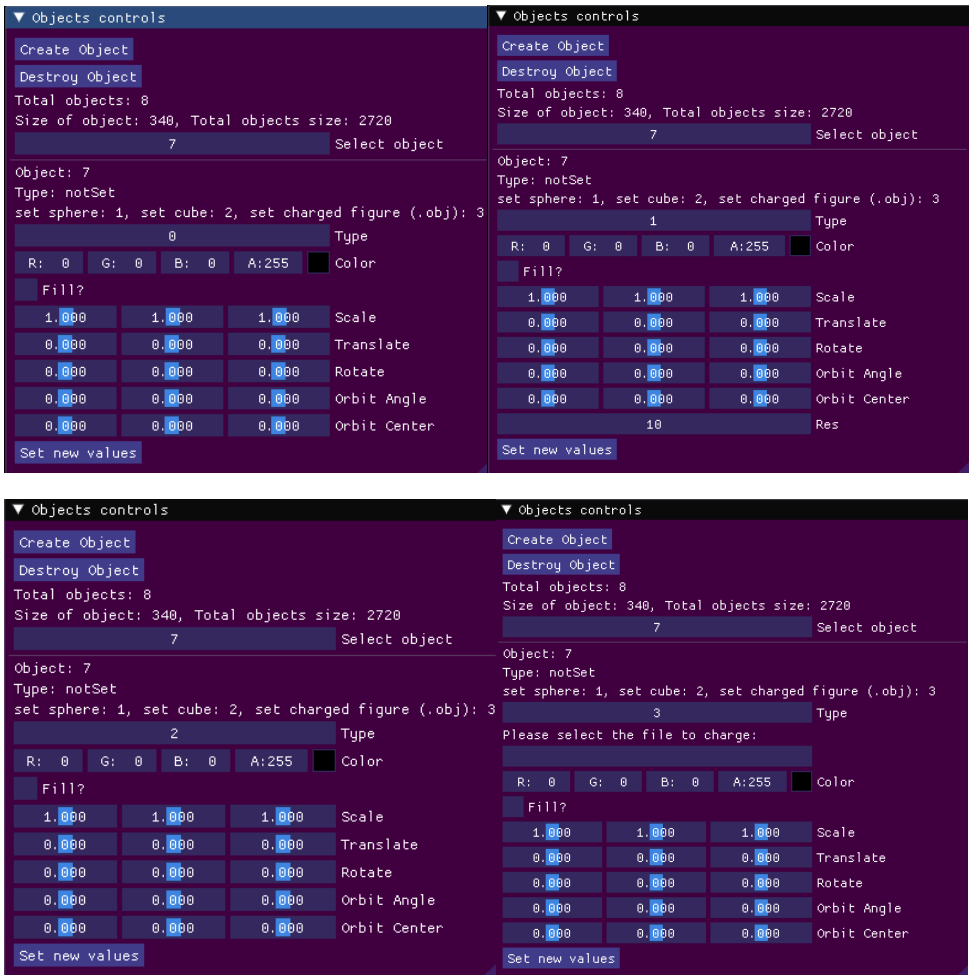
- Fields:

- **Create object:** With this button we can create an undefined object, which will be given attributes later.
- **Destroy object:** With this button we can destroy the number of the selected object. If you destroy every object you will see that window.



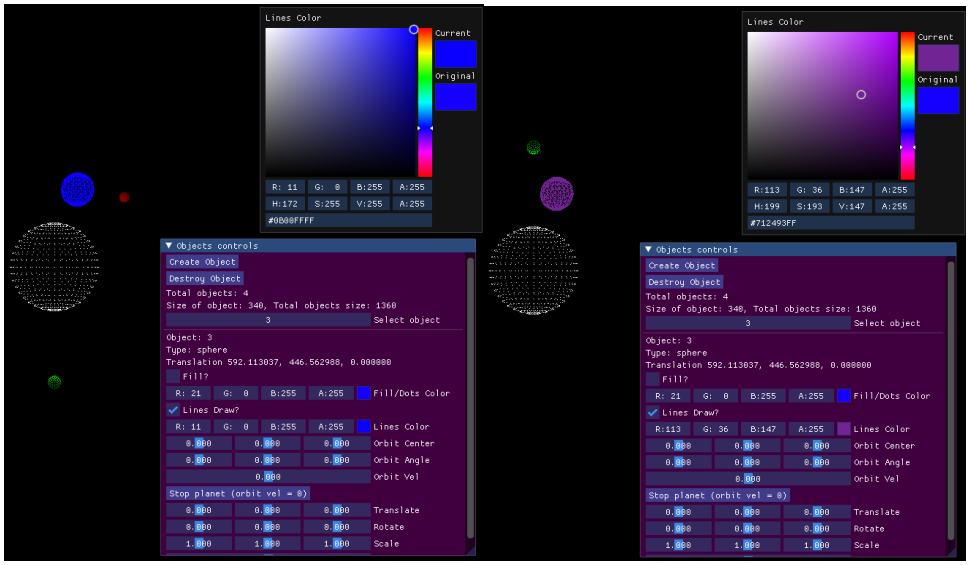
- **Total objects:** It shows you the total number of objects on the screen.
- **Size of object:** Shows you the size of an object.
- **Total objects size:** Shows you the total size of all objects on the screen.
- **Select object:** With this field you can select the object you want to edit.
- **Object:** Shows you the selected object.
- **Type:** It tells you the type of object selected.
- **Translation:** It tells you the location of the selected object.
- **Fill:** With this field you can fill in the selected object.
- **RGBA:** With this field you change the color of the object.
- **Lines draw:** With this field you draw lines between all the vertices of the same face.

- **Orbit:** With this you can edit the orbit of the object, its center, its orbit and its velocity.
- **Stop planet:** With this button you can make the selected object stop.
- **Translate:** You can edit the position of the object separately.
- **Rotate:** You can edit the rotation of the object separately.
- **Scale:** You can edit the scale of the object separately.
- **Equal scale:** You can edit the scale of the object with all 3 axes at the same time.



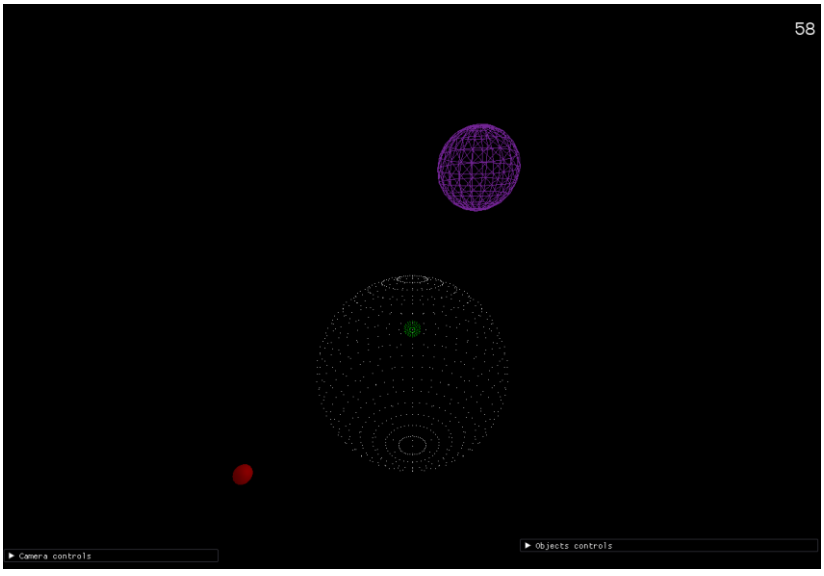
When you click on Create object mos, the first image will appear with a 0 in the type box.

In type 1 we have the field res which is the resolution of the sphere, in type 2 it is the generic one and in type 3 we have a text field where we are told to write the path of a 3d object or an .obj (we have to be careful because we can put any text file but if it is one that is not a 3d object, .obj, the programme could not load the data correctly).



The Fill/Dots color field is to set the color if you use fields or only dots. You can put the color in rgb or by clicking the colored square it appears in another window in which you can see the selected color.

The Lines Color is for lines and Fill/Dots is for fill. But if you don't want fill or lines. The object is going to be drawn by dots and the color of Fill/Dots field



Here we see another perspective of the system. That is because we move the camera with the camera controls.

6º Conclusions & Future Work

Great care has been taken in every detail. There are no memory leaks, no blind spots where objects are not rendered well with respect to the camera. All lights are calculated according to their position with the position of a directional light. And of course they are drawn in order to prevent objects behind from being drawn on top of others etc. In addition, the alpha blender has been added as a special detail.

Problems:

- Additionally, the far and near of the camera can only be modified in its original position, since if it is done after modifying its other values it is out of line with reality.

- I would also love to optimize the code or even use multi threads for the calculation of points (Since it is what consumes the most execution time).