Hello!

Many people like to play computer games. Some prefer platformers, some racing, FPS, MMO, etc.

But few people ever learn to program, usually because it seems to be difficult and time consuming

I will try to show you that creating games is quite simple and can be a lot of fun. In which you basically take the role of God in creating your own digitally interactive worlds.

First, we need to pick out a few tools to use and a game engine that will run our game. There are a lot of popular rapid application development tools for games including ENIGMA, Unity3D, and Blitz Basic.

- Game Editor
Game Editor is a simple point and click tool, which you need to know English to use. While most things can be easy to use, then you need to know which to choose. The project itself is licensed under GNU GPL and can convert a number of platforms (mostly old phones and basic free operating systems (although you can try to compile on BSD or other systems)

- Enigma - dev (I'm going to use it)
Enigma is an intermediate program (if you can call it that) - In general, amateurs can create games as well as advanced users (of course, the more we can, the more we create miracles) , based on EDL (Enigma Development Language). In general, you can call the program a clone of Game Maker (the developers generally try to achieve compatibility)

- SandBox Game Maker (entering into deep water , resort)
Although I tried to learn it - I failed. SBGM used to create RPG, FPS and platforming games in 3D (on the Cube 2 engine), generally everything looks brilliant and as it was being mastered, it has to beat a lot of open source games (and of high quality , something I have to play the game)

1) Planning
The first thing we should do is plan a little bit about the game. To start we will create a basic game since you are simply starting out and do not want a lot of complexity yet.

The second thing is graphics, as games require a lot of high quality art and animation. The problem here is that usually developers and coders are not very good artists and rely on others talents. Another problem is finding a suitable paint program, there are a lot of popular free ones such as Paint.net, which is preferred by ENIGMA's lead developer Robert, as well as Gimp which is available cross-platform. You can also find a lot of free graphics content on the web, You should take a look at : http://www.opengameart.org.

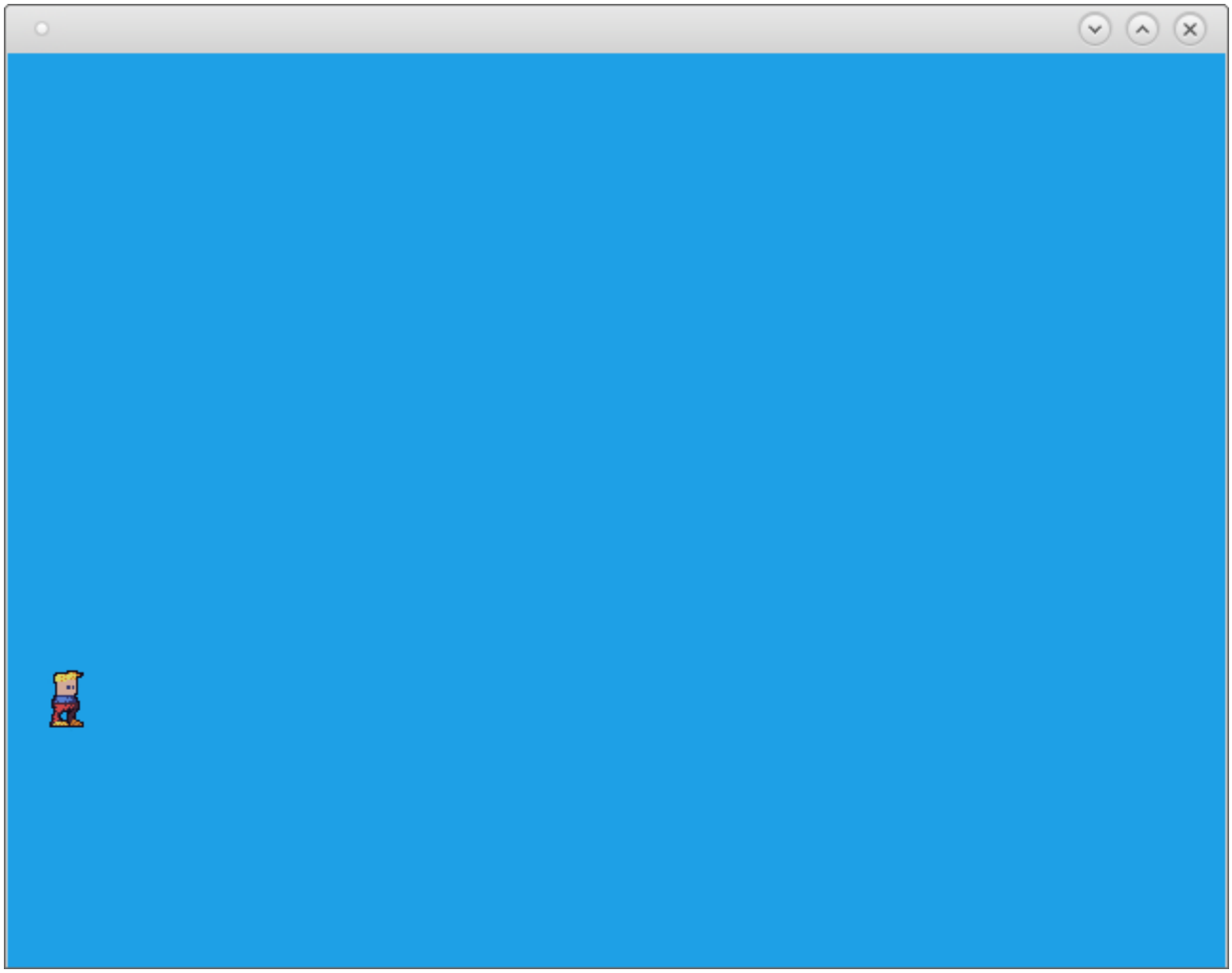
Audio and Video - these two are usually a big problem for indie developers without money to hire musicians and artists, you can of course use free and open source media found on many popular websites, just so long as you be careful about copyright infringement. However usually games that do have their own custom content can be a lot more appealing.

2) Picking Names
It is worth noting that "my game" is just an example, or a prototype, if you will, so it is not meant to be a full blown platformer.

http://opengameart.org/content/outside-tileset
I chose this tileset because it is simple and we can quickly take advantage of it.

1) Character Control
First, we should start with the main character, draw it in Paint (I use the Pinta, but KolourPaint is also good), create a new Sprite, paste it and name the resource properly. Since resources are actually variables they should and must follow the same naming conventions, i.e. do not add weird characters or spaces, underscores are fine, and do not make the first character a number. The standard is usually prefixing sprites as spr_* and objects as obj_*, etc. It is also more sensible to name the objects and other resources what they represent such as a player object would be called obj_player, etc.

You should create an object (obj_hero) and assign an image (spr_hero), create a new room and place the instance there.



2) Movement
Each character in the game should not move, unless of course you are in a sort of spectator mode. We will move to the right, left, up and add a little bit of gravity.
It is helpful if you already know basic highschool geometry and math, for instance the Cartesian coordinate system is used to map objects and graphics on the screen as well as calculate collision detection and make physics calculations. The character object we need to add events to handle the input from the arrow keys and move it accordingly.

We move the object to the right using this code:

```
x=x+3
```

This sets the x coordinate of the object equal to its current position in addition to 3 pixels. To move left you simply subtract from x.

For jumping we need to add a little more code than what we did for moving left or right. To move the object vertically without accounting for gravity, such as walking up a ladder you would use the following code.

```
if !place_free(x,y+1)
vspeed=-8;
```

Basically this code says that if no objects are on top of each other the vertical speed becomes -8 pixels per time step.

Now we can add an extra component, gravity, which will gradually pull the object down. You can add this code to the step event of the object.

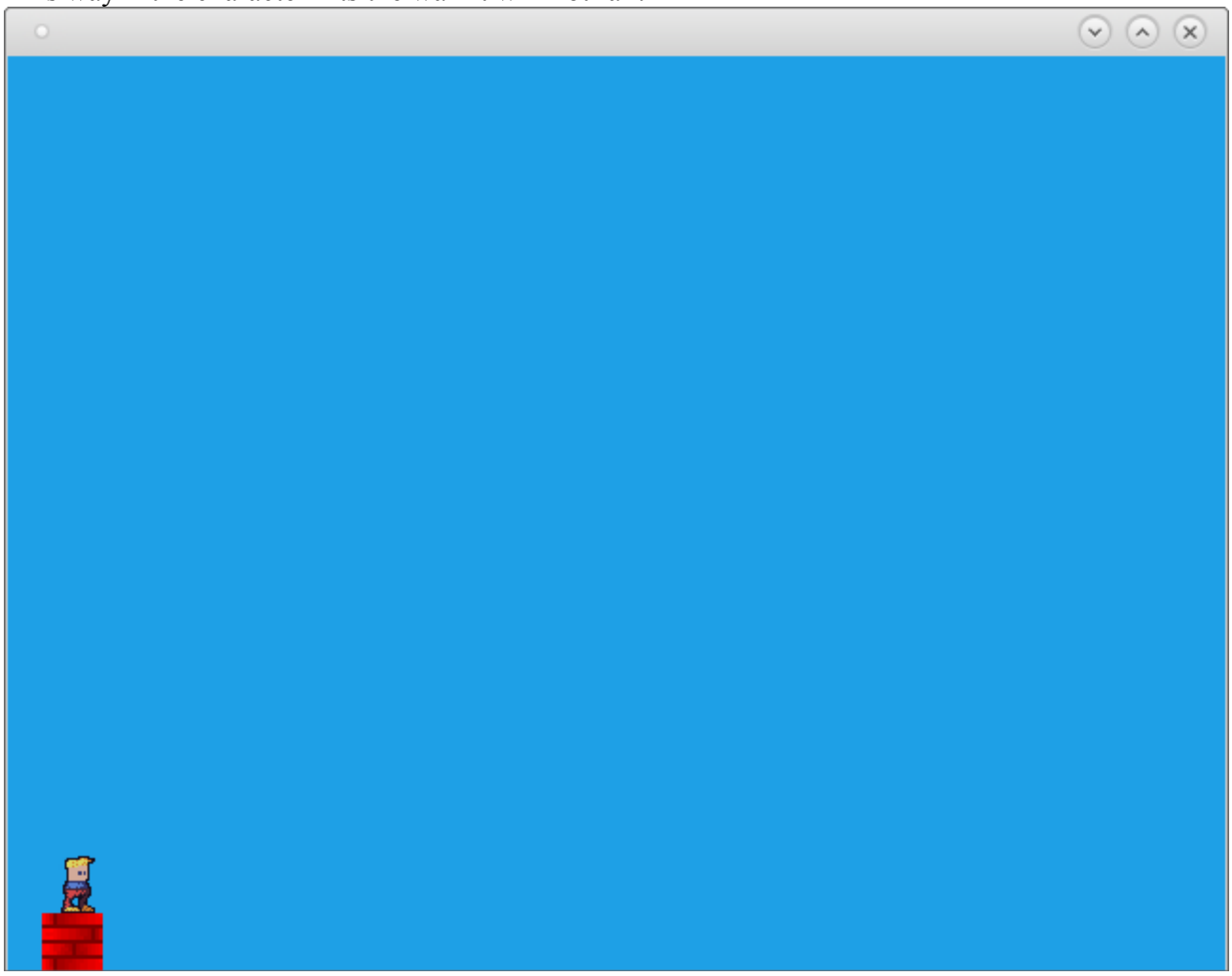
```
if place_free(x+0,y+3) {
  gravity=0.25 }
 else {
  gravity=0
}

if (vspeed>8)
{
  vspeed=8
}
```

We will also need to create an invisible wall for detecting if the object falls of the screen.
Add an object and tick the solid option on it, which is generally for stationary objects.

In obj_hero add a event Collision with obj_wall and enter the code:

```
if (vspeed> 0) {
  move_contact_solid (270, vspeed);
}
vspeed = 0;
```

This way if the character hits the wall it will not fall.



3) Drowning and Obstacles

The figure we are using from our tileset includes water which the player can not swim in. First cut the water out of the tileset and add it to the sprite and create an object called obj_water.

I chose the darker water (it does not make any difference)

Now create a collision of obj_water and add the code

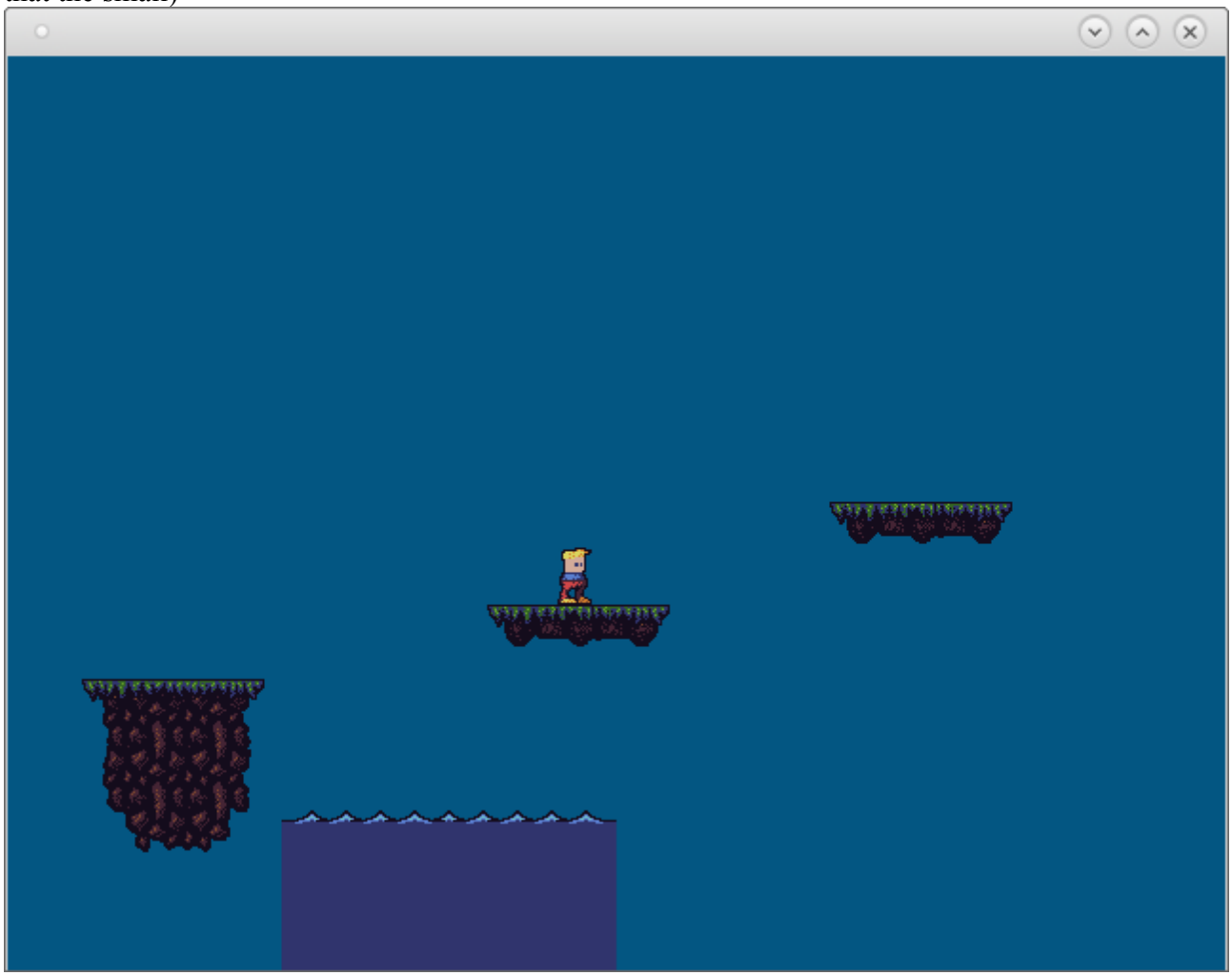
```
game_restart();
```

That is to say that the game (including our position, and all other variables) will restart, almost as if it just started.

4) Land
This is where we check for obj_wall with spr_wall and replace the soil. Specifically, we throw spr_wall, add spr_ground in obj_wall change the sprite to spr_ground, and change the name of the object itself. Phew

5) Background
The tileset did not give us the background, so either we'll get it, or we use a uniform color. I personally like this bright blue like the default in the settings of the room editor (in the Background tab) changed to a darker color.

6) A small earth
Adding even object " small world", which will be the platform. There is nothing like the earth. (so that the small)



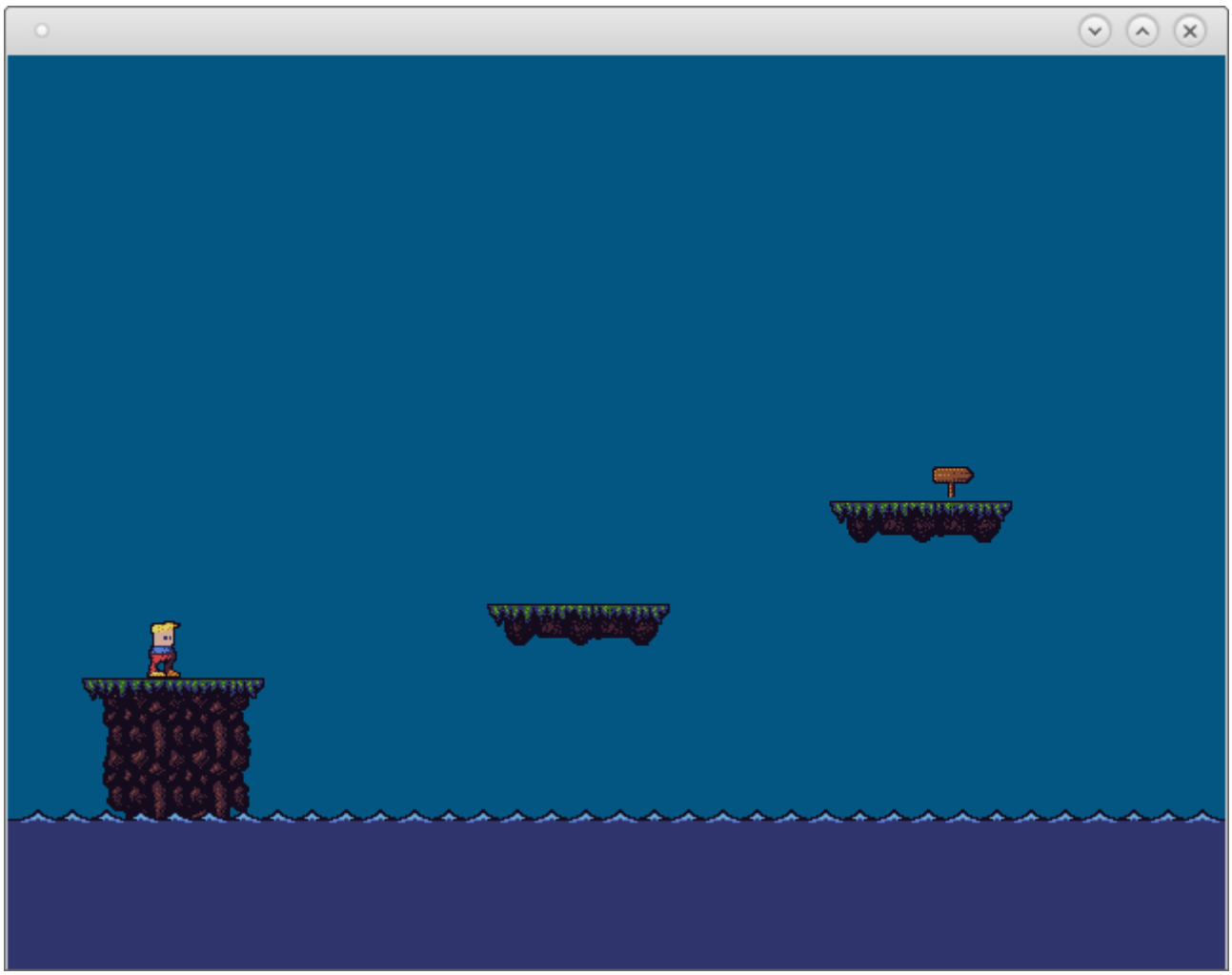
7) End of game
Everything has a beginning and an end - we need to add an end. Employ the graphic character (with an arrow , but whatever) as our goal.

Here we add a character, add an object with the graphics of the character in the form of adding a collision with a sign. This time, our code is:

```
game_end (); // turn off the game
```

8) The amendments visual
To our game look nice , we need to improve it visually - let the water covers the entire territory , etc.

Voila! The game is over!



Of course we have many mechanisms introduced in the game, such as the prevention of possible escape from the game. But it's in your hands what you do. My role has come to an end, I can die in peace :)

Useful links:
http://enigma-dev.org/
Official ENIGMA Website

http://gmclan.org/up6184_3_gra.html
The source code for our game