ETH Game Programming Laboratory

Interim Report
April 11th, 2011

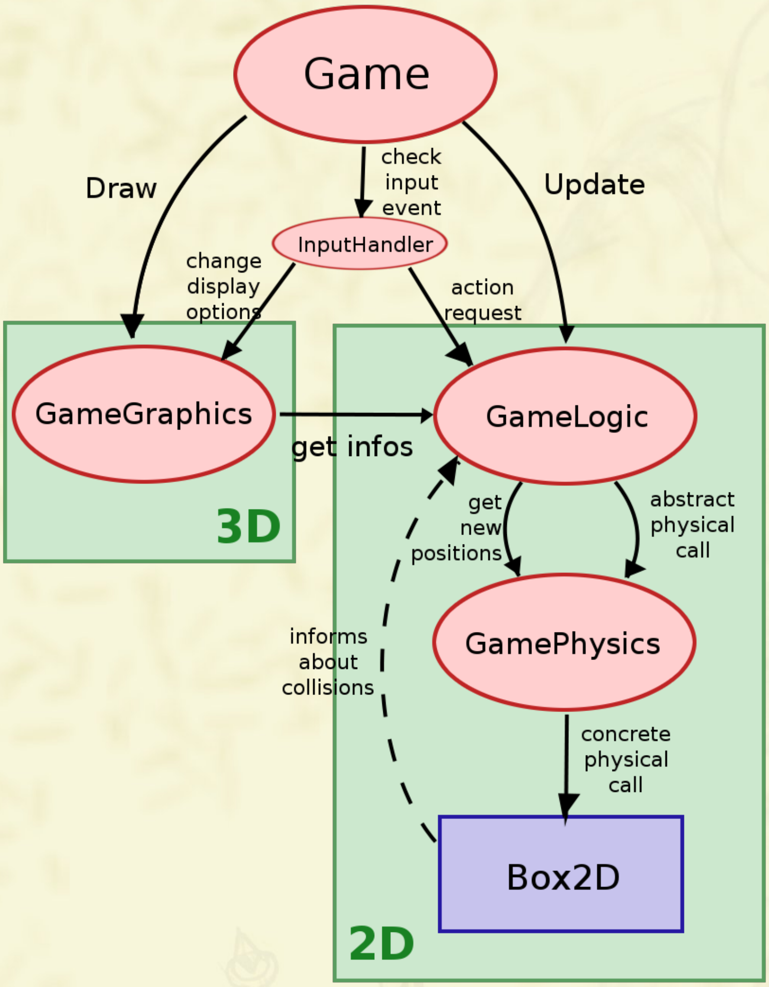
Ra.M.Bo Studio

Code Architecture

First, even if it isn't a feature visible in the game, the first completed task was to turn ourselves into software engineers, to create a good architecture that makes easier the further development.

The big picture is the following, you can have more details on:

https://twiki.graphics.ethz.ch/GameClass/TeamRaMBoArchitecture



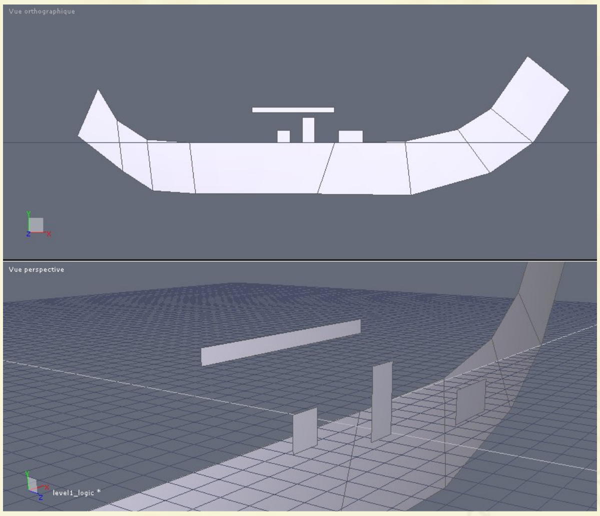
Physics Engine

We decided to use Box2D as the physics engine, because of its simplicity (we do not need complex physics effects).

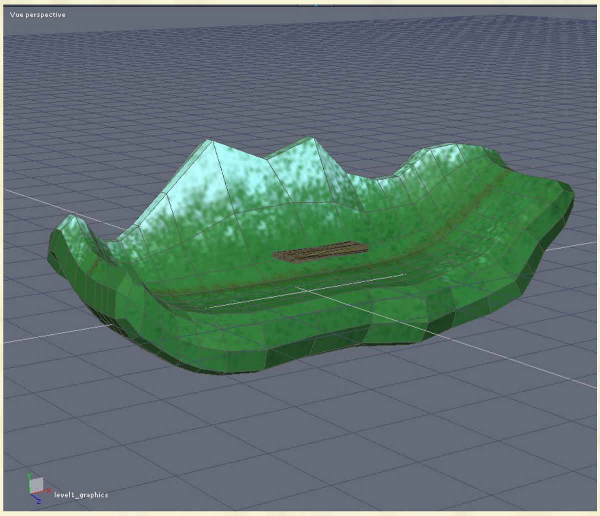
The features currently implemented are:

- Characters and enemies as polygonal dynamic bodies
- Magical balls as circular dynamic bodies
- Modyfying the source code of Box2D, adding a new attribute to dynamic bodies to decide if gravity should be taken into account or not (magical balls are not subject to gravity, for instance)
- A level importer, to convert a fbx file to Box2D and Logical objects. The fbx model is basically a 2D mesh, where the z coordinate is used to add informations. (this is part of the map, this is an enemy, ...)
- Methods to make the characters move, jump and attack.
- Advanced collision detection to infer the 'current state' of a character (is he waiting, jumping, falling, sliding, ...)

2D model imported for the logic part:



3D model imported for the graphics part:



Misc. section

Camera control

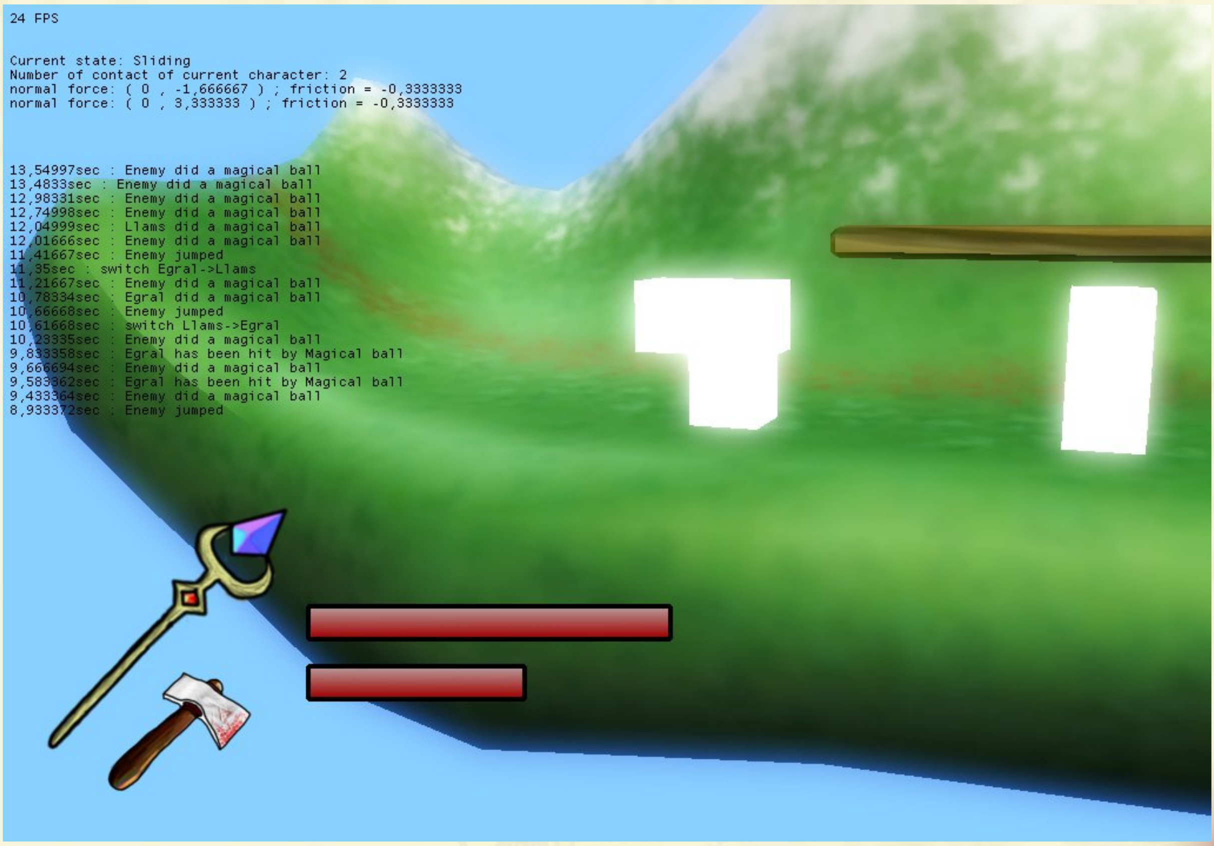
The camera is automatically moved in a smooth and intelligent way: views the two characters if they are close enough, otherwise only follows the current character.

Artificial Intelligence

The enemy moves and attacks automatically. However, at this moment, we can't really say it is an "intelligence", since only random actions are used...

Head-Up Display

The health of Characters can be displayed.
We implemented a little printing system to debug easily.



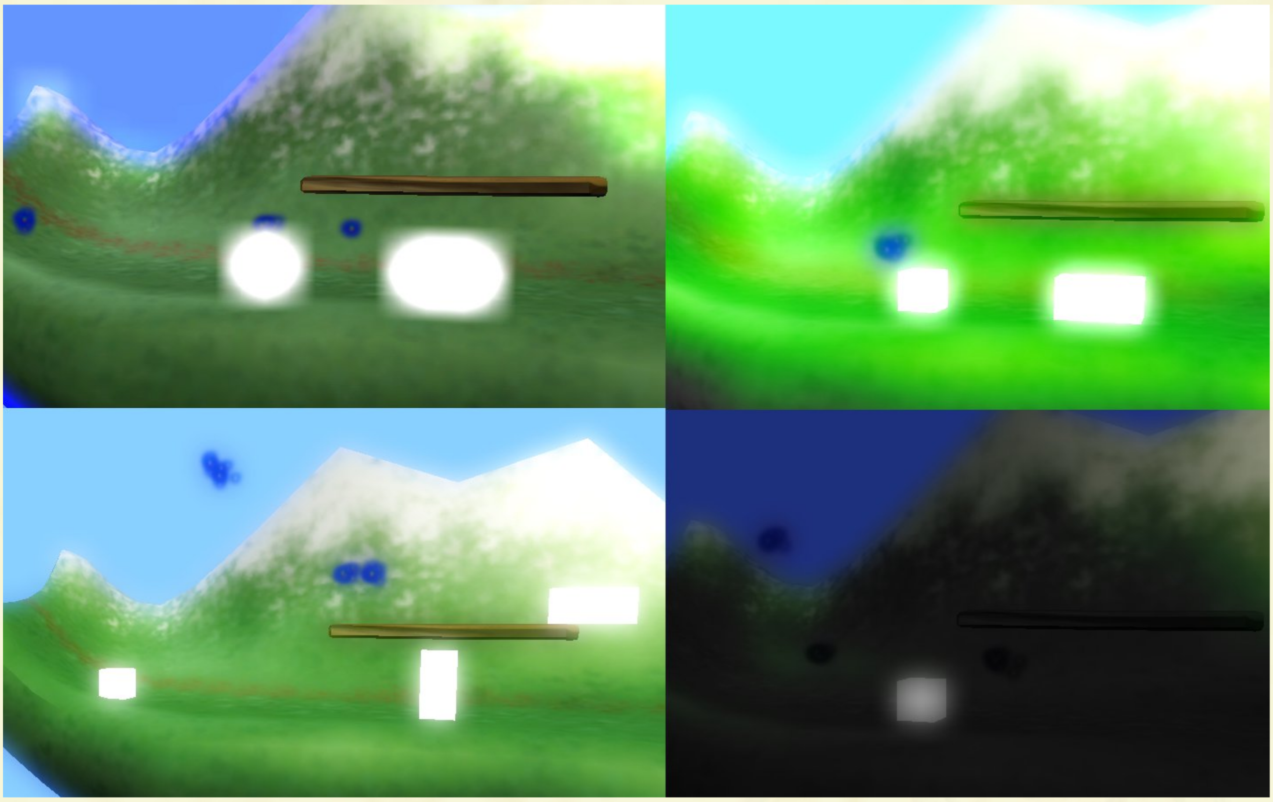
```
24 FPS

Current state: Sliding
Number of contact of current character: 2
normal force: ( 0 , -1,666667 ) ; friction = -0,3333333
normal force: ( 0 , 3,333333 ) ; friction = -0,3333333


13,54997sec : Enemy did a magical ball
13,4833sec : Enemy did a magical ball
12,98331sec : Enemy did a magical ball
12,74998sec : Enemy did a magical ball
12,04999sec : Llams did a magical ball
12,01666sec : Enemy did a magical ball
11,41667sec : Enemy jumped
11,35sec : switch Egral->Llams
11,21667sec : Enemy did a magical ball
10,78334sec : Egral did a magical ball
10,66668sec : Enemy jumped
10,61668sec : switch Llams->Egral
10,23335sec : Enemy did a magical ball
9,833358sec : Egral has been hit by Magical ball
9,666694sec : Enemy did a magical ball
9,583362sec : Egral has been hit by Magical ball
9,433364sec : Enemy did a magical ball
8,933372sec : Enemy jumped
```

Post-Processing Shader

A post-processing 'bloom' shader has been implemented.

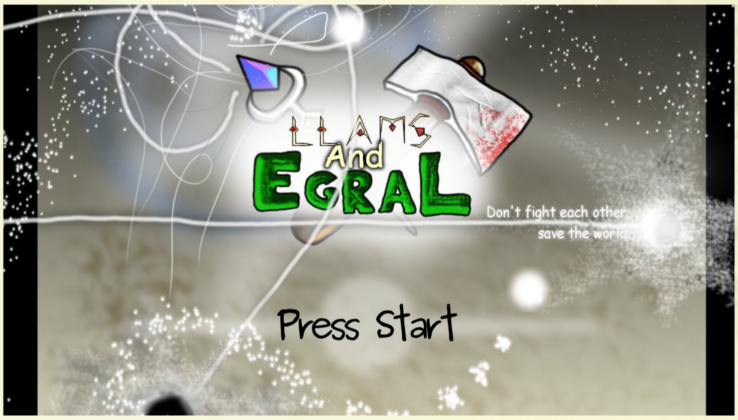


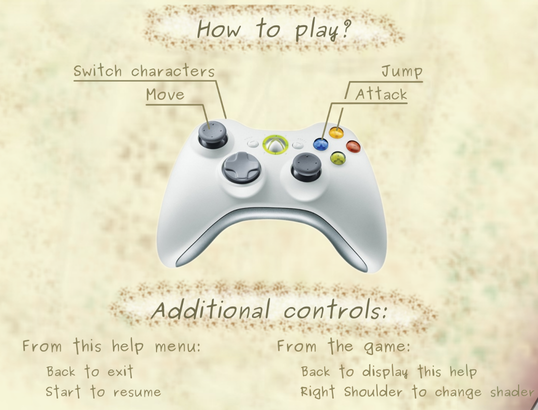
Particle Systems

A 3D particle system has been implemented for the magical ball (see above), and a 2D particle system for the Startup Menu (see below)

Startup and Help Menus

To make the game looks like more a real game, we added a little animated startup menu, and the required help menu to learn how to play





Additional controls:

From this help menu:

Back to exit

Start to resume

From the game:

Back to display this help

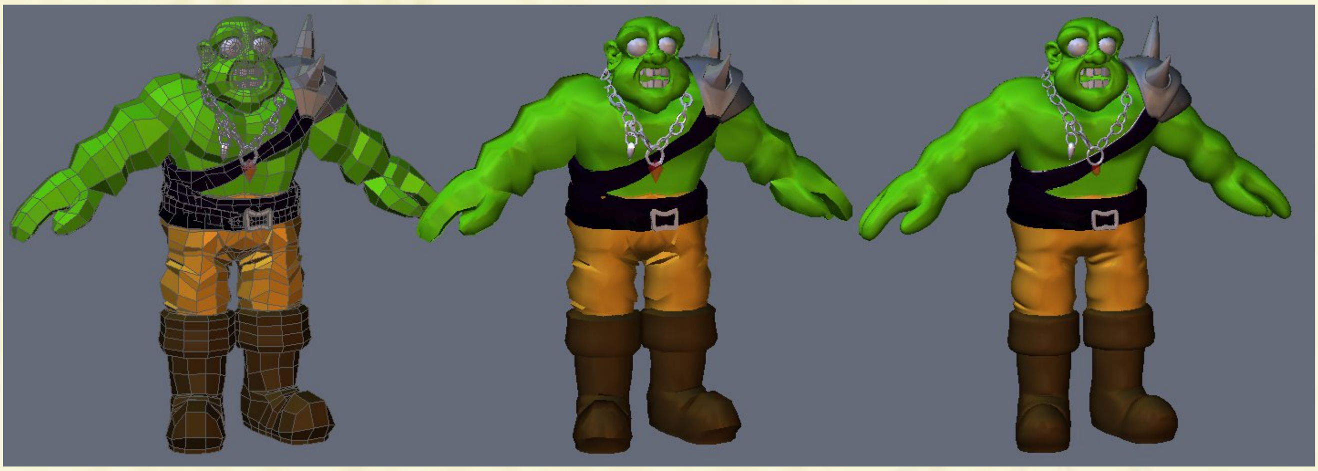
Right Shoulder to change shader

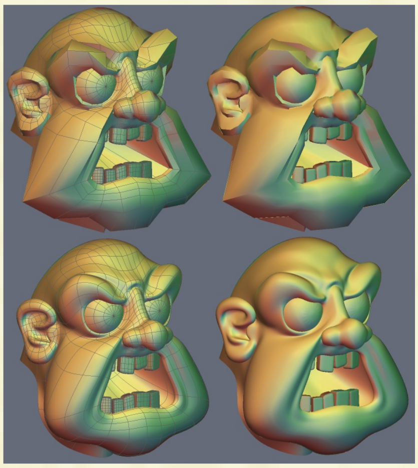
Sounds

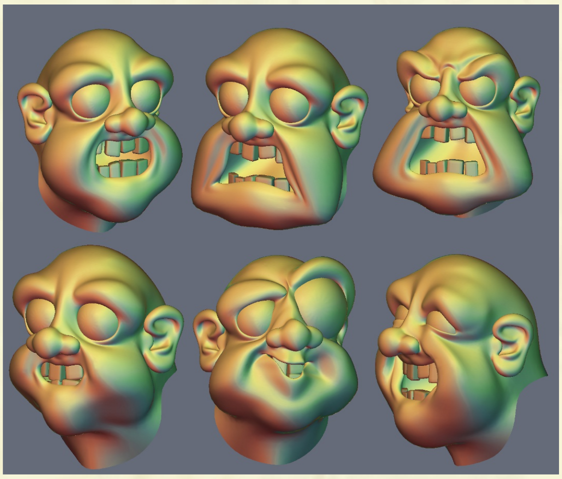
Some sounds has been added to the game, to enhance the immersive power of the game.

3D modeling

A very simple level has been modeled, to test the gameplay (see page 3)
Egral has been modeled

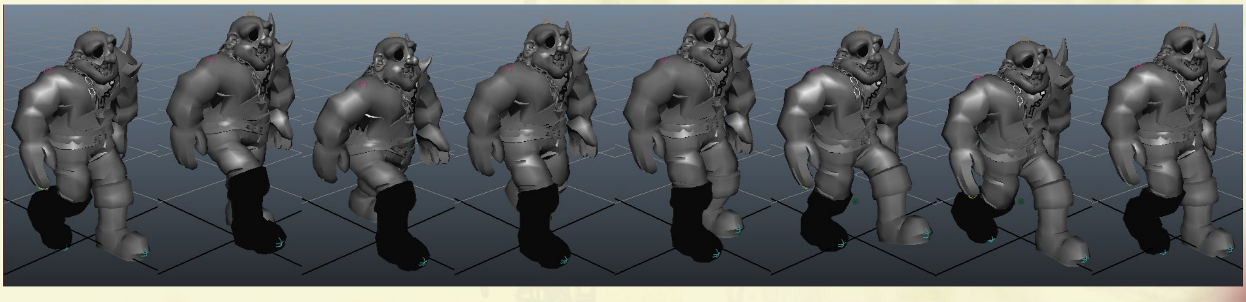






Character animations

A simple walking animation have been created for Egral, and is currently being integrated on the game.



Conclusions

We can consider that we are currently implementing the Desired Target, since almost all the Low Target is complete (there is only some 3D modeling that are missing, and the animation framework is almost in place), and some features of the desired target or more are already at least partially implemented (Startup menu, Particle system, shader, Level importer...)

This means that we are in advance according to our timeline in the game proposal. Thanks to this, we could probably have more features for the alpha release than expected, which is really motivating.

The level importer has been implemented far earlier than planned, because it appears it would be better to test right now the gameplay on real - but simple - modeled levels. The animations, even if there are not totally implemented yet, appear to be more easily implementable than expected: we didn't know that the Shawn's "SkinnedMesh" existed. Then, it will be possible to spend more time on other graphical effects.

On the contrary, implementing the ball attack was more time-consuming than expected, since it has been necessary to modify the physics engine. In the same way, preventing a character to jump again when he is still in the air was a problem we didn't initially think about, and that has required to go deeper than expected in the Box2D functionnalities, to get precise informations about the collisions.