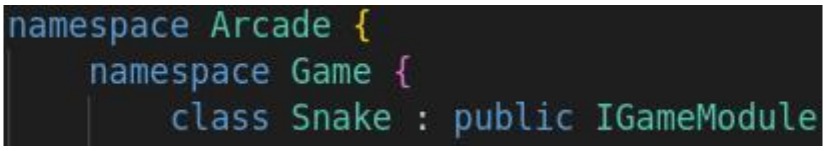
How To : Implement a new game lib

1) Create a new class that inherits from IGameModule



```
namespace Arcade {
    namespace Game {
        class Snake : public IGameModule
```

2) Implement IGameModule's methods

• std::pair<int, int >getResolution () const

Get the Resolution of the game.

• void start ()

Start the game.

• void end ()

End the game.

• const size_t & getScore () const

Get the Score.

• void setInputs (std::vector<Arcade::Graphics::IDisplayModule::Events >e)

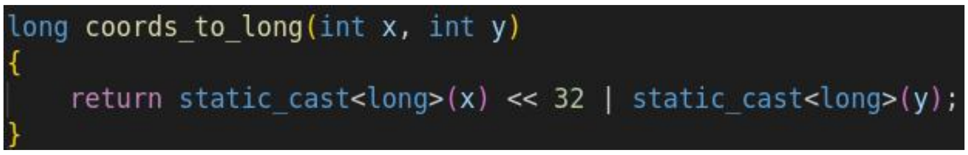
Set the User's Keyboard Inputs.

• const std::unordered_map<long, unsigned int >& getPixels () const

Returns a map of pixels to display.

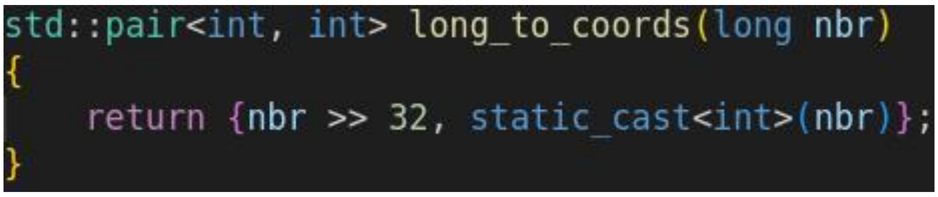
<long position, unsigned int color>

Use this to compress x and y coordinates to a long :



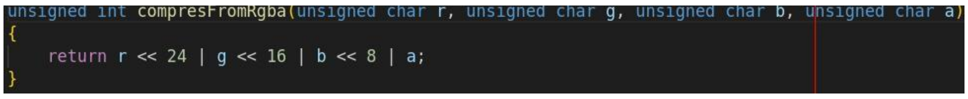
```
long coords_to_long(int x, int y)
{
    return static_cast<long>(x) << 32 | static_cast<long>(y);
}
```

And decompress it this way :



```
std::pair<int, int> long_to_coords(long nbr)
{
    return {nbr >> 32, static_cast<int>(nbr)};
}
```

Use this to compress an RGBA color to an int ;



```
unsigned int compresFromRgba(unsigned char r, unsigned char g, unsigned char b, unsigned char a)
{
    return r << 24 | g << 16 | b << 8 | a;
}
```

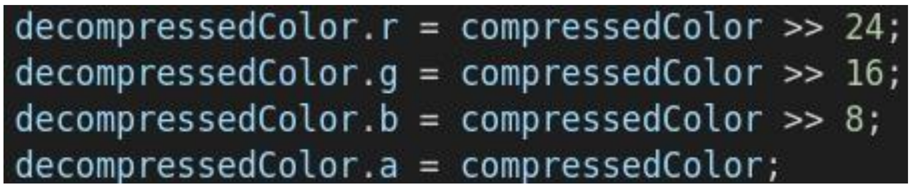
R : Red color intensity (0 – 255)

B : Blue color intensity (0 – 255)

G : Green color intensity (0 – 255)

A : Alpha (0 – 255)

And decompress it this way :



```
decompressedColor.r = compressedColor >> 24;
decompressedColor.g = compressedColor >> 16;
decompressedColor.b = compressedColor >> 8;
decompressedColor.a = compressedColor;
```

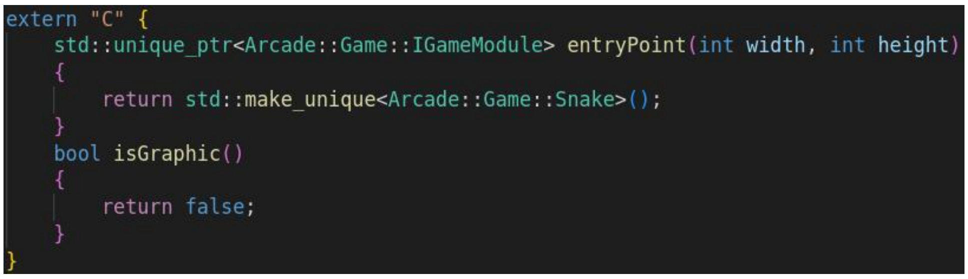
• void refresh ()

Refresh the game, here goes your game engine.

• void reset ()

Restart the game.

3) Add entryPoint and isGraphic functions



```
extern "C" {
    std::unique_ptr<Arcade::Game::IGameModule> entryPoint(int width, int height)
    {
        return std::make_unique<Arcade::Game::Snake>();
    }
    bool isGraphic()
    {
        return false;
    }
}
```

EntryPoint is called by the Arcade’s Core and needs to return an instance of your game class

IsGraphic returns false

4) Compile your game lib

g++ -o lib/arcade_${GameLibName}.so ${GameLibFiles} -shared -fPIC