VG ENGINE 101

Tutorial

16.10.2015

Contents

GameObjects

Include

#include “engine/game/gameObject.h”

Creation

Example of creating a GameObject named “Foo”:

```
GameObject *Foo = new GameObject(“Foo”);          // Creating GameObject
Scene mScene = new Scene();                       // Creating Scene (if not already created)
mScene->getObjectPool()->addGameObject(test);     // Adding GameObject to the scene
```

See “Components” section on how to add components for your GameObject.

Components

Drawable component

Include

```
#include “engine/game/quadrangleComponent.h”      // For drawable quadrangles
#include “engine/game/triangleComponent.h”        // For drawable triangles
```

Creation

With texture:

```
// Creating quadrangleComponent    with the texture “test.png”
QuadrangleComponent *quadre = Game::getInstance()->getFactory()-
>createRenderComponent<QuadrangleComponent>(“test.png”);

// Creating triangleComponent with the texture “test.png”
TriangleComponent *triangle = Game::getInstance()->getFactory()-
>createRenderComponent<TriangleComponent>(“test.png);
```

Without texture: Coming Soon™

Remember!

If you create drawable component with texture it is loaded from Asset folder set in game project!

Text Component

Include

#include “engine/game/textComponent.h”

Creation

```
// Creating text component with font & size
TextComponent* Text = game->getFactory()->create(“arial.ttf”, 16u);
Text->setText(“test”);                    // Optional: Modify the text
Text->setColour(0, 0, 255);               // Optional: Modify the color (numbers between 0 and 255)
MyTextObject->addComponent(Text);         // Add textComponent to your GameObject
```

Animation Component

Include

#include “engine/game/animationcomponent.h”

Creation

Example of creating an animated GameObject named “animationObject”.

```
// Create a new GameObject
GameObject *animationObject = new GameObject(“Animation”);

// Create QuadrangleComponent spritesheet for the animated GameObject
QuadrangleComponent *animationComponent = game->getFactory()-
>createRenderComponent<QuadrangleComponent>(“spritesheet.png”);

// Add the QuadrangleComponent to the GameObject
animationObject->addComponent(animationComponent);

// Add TransformComponent for the GameObject so it will be placed somewhere later
TransformComponent *animationTransform = new TransformComponent(Vector2<int>(int positionX, int
positionY), Vector2<int>(int sizeX, int sizeY), float rotation);

// Add the transformComponent to your GameObject
animationObject->addComponent(animationTransform);

// Create and add the animationComponent for your GameObject so it will be animated
animationObject->addComponent(new AnimationComponent(float animationInterval, int rowCount, int
columnCount, int total frameCount));

IMPORTANT!!
// Create and add AnimationSystem for animationComponents to work!
AnimationSystem *animationSystem = new AnimationSystem();
game->addComponentSystem(scene, animationSystem);

// Add the animated GameObject to the scene
scene->addGameObject(animationObject);
```

Physics Component

Include

```
#include "engine/game/physicsSystem.h"
#include "engine/game/physicsPolygonComponent.h"

// Create transform component for physics component
TransformComponent *physicsTransform = new TransformComponent(Vector2<float>(80, 64), Vector2<float>(64, 64), 0.0f);

// Create QuadrangleComponent
QuadrangleComponent *physicsQuadrangle = new QuadrangleComponent("sample.png");

// Create new physics polygon component with dynamic body
PhysicsPolygonComponent *physicsComponent = new PhysicsPolygonComponent(physicsTransform, PhysicsComponent::DYNAMIC, PhysicsSystem::world, 64, 64);
```

NOTE Last 2 parameters are optional, if you don't pass them, physics objects collision will be the same size as its defined in the transform component (same size as texture)

```
// Add physics component to physics gameobject physicsTestObject ->addComponent(physicsComponent);

// Add transform to physics gameobject
physicsTestObject ->addComponent(physicsTransform);

// Add QuadrangleComponent to physics gameobject physicsTestObject->addComponent(physicsQuadrangle);
```

Your Own Components

Example of creating a component called "MyComponent"

MyComponent.h

```
#include <engine/game/component.h>                    //Include the base header
class MyComponent :public vg::Component               //Public to vg::Component
{
public:
        TestComponent();
        ~TestComponent();
};
```

Example of creating a System called "MySystem"

MySystem.h

```
#include "engine/game/system.h"

using namespace vg;
class MySystem : public System
{
ShipSystem();
~ShipSystem();
void update(
};
```

MySystem.cpp

```
#include “MySystem.h”
#include “engine/game/game.h”

using namespace vg;

MySystem::MySystem() :System()
{
// Add your own code here
}

void MySystem::update(std::vector<vg::gameObject*> *gameObjects, float deltaTime)
{
		if ((*it)->getName() == “mygameobject”)
		{
		// Add your own logic here

		}
}
```

Usage

Example of calling your own component in main.cpp

```
MyComponent *myComponent = new MyComponent();
object->addComponent(myComponent);

MySystem *system = new MySystem();     // Remember to include
```

Sound

Include

#include “engine/sound/AudioManager.h”

Creation

vg::sound::Sound* testSound = new vg::sound::Sound(“shoot.mp3”); // Creating a new sound

Usage

Game::getInstance()->getAudioManager()->addSound(*testSound); // Playing the made sound object

Custom Shader

Creation

Place the shader soure files to “ProjectFolder/assets/shaders”.

Usage

Game::getInstance()->getGraphics()->switchShader(“vertex.glsl”, “fragment.glsl”);

Input

Include

```
#include “engine/input/keyboard.h”      // For keyboard
#include “engine/input/mouse.h”         // For mouse
#include “engine/input/sensor.h”        // For android sensors
#include “engine/input/touch.h”         // For android touch
```

Usage

```
input::Keyboard::                      // For keyboard
input::Mouse::                         // For mouse
input::Sensor::                        // For android sensors
input::Touch::                          // For android touch
```

For example:

```
input::Touch::getIsReleased()          // Returns whether touch is being released from the screen
```

Camera

Include

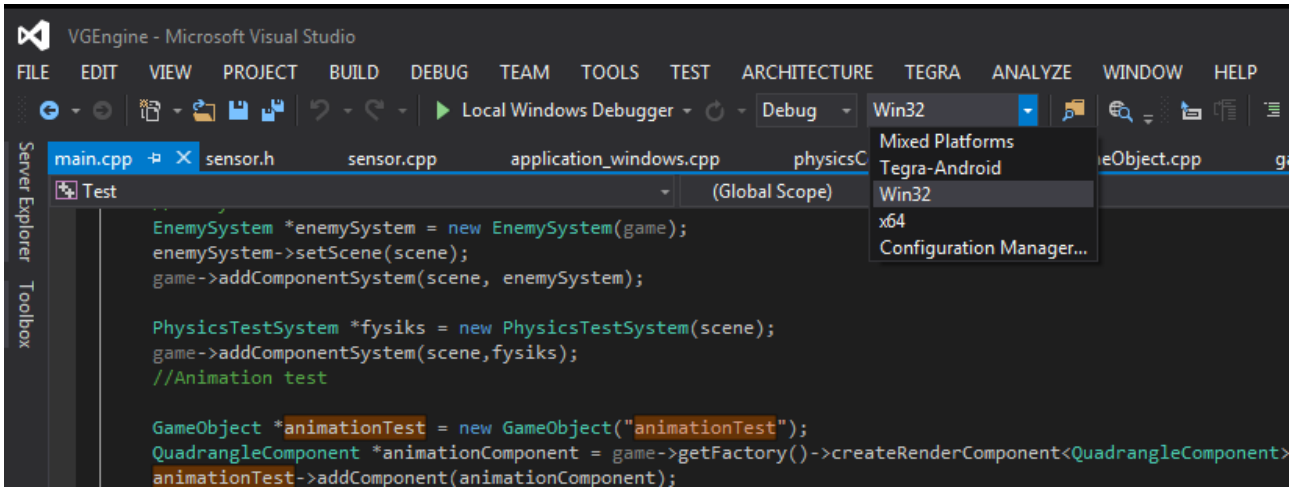
Screen

Include

Windows Version

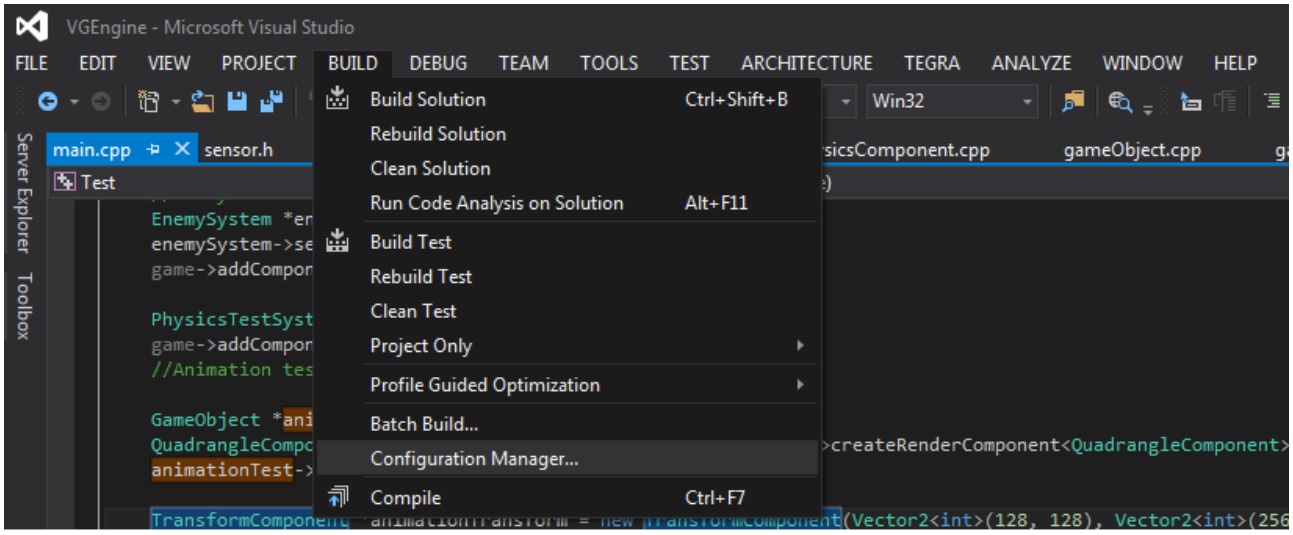
Usage

Select Win32 as solution platform.



If Win32 doesn't appear, do the following:

Build -> Configuration Manager



Active solution platform -> Choose “Win32” and then Press Close