Pong.h

```
#pragma once

#include "GE161.h"
#include "Game.h"

class Pong : public GE161::Game
{
public:

    // All game classes must override these two Game methods.
    void setup();
    std::string chooseScene(std::string prevScene, int prevReturnCode);

};
```

Pong.cpp

```
//An object of the class derived from GE161::Game must be created.
Pong* theGame = new Pong();

// setup() is called once, before the window is created or displayed.
void Pong::setup()
{
    window()->size(500, 350);
    window()->title("Pong");

    addScene("Instructions", new PongInstructionsScene(this));
    addScene("GamePlay", new PongGamePlayScene(this));
}
```

Pong.cpp

```
std::string Pong::chooseScene(std::string prevScene, int prevReturnCode)
{
    if (prevScene == START_GAME)
    {
        return "Instructions";
    }

    if (prevScene == "Instructions")
    {
        return "GamePlay";
    }

    // prevScene must == "GamePlay"
    return EXIT_GAME;
}
```

PongInstructionsScene.h

```h
#pragma once

#include "GE161.h"
#include "Pong.h"

class PongInstructionsScene : public GE161::Scene
{
public:
    PongInstructionsScene(Pong* theGame);
    ~PongInstructionsScene();

    bool setup();        // Override Scene methods
    int draw();

    void onEvent(GE161::Event& ev); // Override EventListener method.

private:
    Pong* theGame;
    GE161::GameObject* instructions;
    int frameIndex;
    bool playGameRequested;
    bool exitRequested;
};
```

PongInstructionsScene.cpp

```
#include "GE161.h"
#include "PongInstructionsScene.h"

PongInstructionsScene::PongInstructionsScene(Pong* game) :
theGame(game)
{
}

PongInstructionsScene::~PongInstructionsScene()
{
    delete instructions;
}
```

PongInstructionsScene.cpp

```
bool PongInstructionsScene::setup()
{
    instructions = new GE161::GameObject(20, 20);
    GE161::Sprite* instructionsImage = new GE161::Sprite(450, 300);
    frameIndex = instructionsImage->makeFrame(
            GE161::Game::basePath() + "Instructions.png", 0, 0);
    instructions->attachSprite(instructionsImage);

    playGameRequested = false;
    exitRequested = false;

    GE161::Game::registerAsListener(GE161::Event::KEYBOARD_EVENTS, this);

    GE161::Game::debugOut("PongInstructionsScene::setup completed!");
    return true;
}
```

PongInstructionsScene.cpp

```
int PongInstructionsScene::draw()
{
    instructions->draw(frameIndex);

    if (playGameRequested)
    {
        return 1;        // don't call draw again, go on to next scene
    }
    else if (exitRequested)
    {
        return -1;       // stop program
    }

    return GE161::Game::CONTINUE_SCENE;
}
```

PongInstructionsScene.cpp

```
void PongInstructionsScene::onEvent(GE161::Event& event)
{
    if (event.type == GE161::Event::KEY_DOWN)
    {
        if (event.key == " ")
            playGameRequested = true;
        else if (event.key == "X")
            exitRequested = true;
    }
}
```

PongPlayGameScene.h

```
#pragma once

#include "GE161.h"
#include "Pong.h"

class PongGamePlayScene : public GE161::Scene
{
public:
    PongGamePlayScene(Pong* theGame);
    ~PongGamePlayScene();

    bool setup();
    int draw();

private:
    Pong* theGame;
    GE161::GameObject* ball;
    int frameWidth_;
    int frameHeight_;
    int bounceCount_;
    int x_delta, y_delta;
};
```

PongPlayGameScene.cpp

```
#include "PongGamePlayScene.h"

#include <stdlib.h>      // srand, rand
#include <time.h>        // time


PongGamePlayScene::PongGamePlayScene(Pong* pong) :
theGame(pong)
{
}


PongGamePlayScene::~PongGamePlayScene()
{
    delete ball;
}
```

PongPlayGameScene.cpp

```
bool PongGamePlayScene::setup()
{
    // Rolling ball sprite sheet is from
    //  http://www.java-gaming.org/index.php?topic=21935.0.
    // It has 32 64x64 frames.

    // Start the ball with its upper left in the middle of the window.
    ball = new GE161::GameObject(
        theGame->window()->clientWidth() / 2,
        theGame->window()->clientHeight() / 2);
    frameWidth_ = frameHeight_ = 64;
    GE161::Sprite* ballSpriteSheet =
                new GE161::Sprite(frameWidth_, frameHeight_);
    for (int row = 0; row < 4; row++)
    {
        for (int column = 0; column < 8; column++)
        {
            int f = ballSpriteSheet->makeFrame(
                GE161::Game::basePath() + "RollingBall.png",
                column*frameWidth_, row*frameHeight_);
            ballSpriteSheet->addFrameToSequence("Rolling", f);
        }
    }
```

PongPlayGameScene.cpp

```
    ball->attachSprite(ballSpriteSheet);

    x_delta = y_delta = 1;
    bounceCount_ = 0;

    // seed the random number generator with the current time
    srand(static_cast<unsigned int>(time(NULL)));

    return true;
}
```

PongPlayGameScene.cpp

```
int PongGamePlayScene::draw()
{
    ball->moveX(x_delta);
    ball->moveY(y_delta);
    ball->draw("Rolling");

    // If the ball has hit an edge, bounce randomly.
    if (ball->getX() < 0)
    {
        x_delta = rand() % 3 + 1;
        bounceCount_++;
    }
    if (ball->getY() < 0)
    {
        y_delta = rand() % 3 + 1;
        bounceCount_++;
    }
    if (ball->getX() + frameWidth_ >= theGame->window()->clientWidth())
    {
        x_delta = -(rand() % 3 + 1);
        bounceCount_++;
    }
```

PongPlayGameScene.cpp

```
    if (ball->getY() + frameHeight_ >= theGame->window()->clientHeight())
    {
        y_delta = -(rand() % 3 + 1);
        bounceCount_++;
    }

    if (bounceCount_ < 10)
    {
        return GE161::Game::CONTINUE_SCENE;
    }
    else     // bounced out
    {
        return -1;  // stop game
    }
}
```