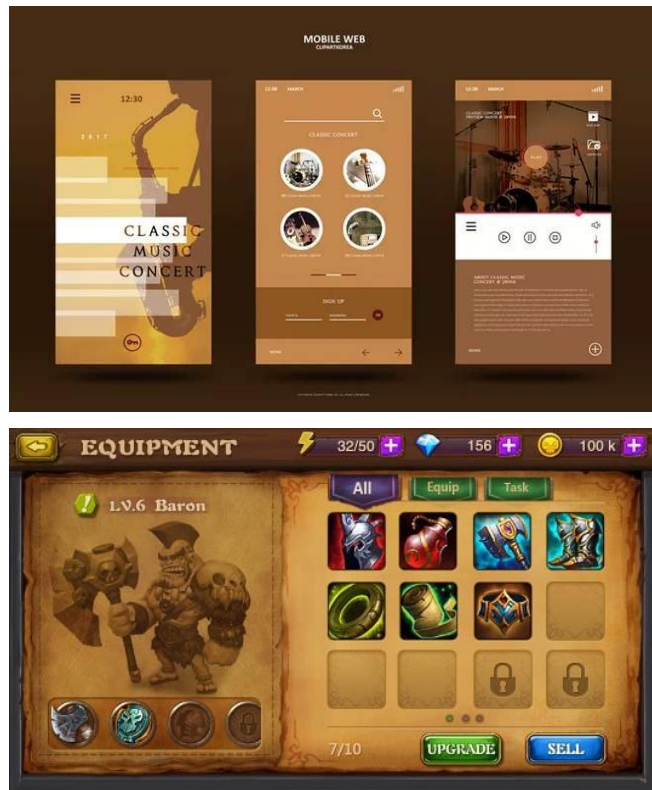


第六章 游戏交互界面设计

上节回顾

• Chapter 6

- 交互界面设计概述
- 游戏的可玩性与交互界面
- 交互界面设计基础
- 游戏软件的交互界面设计
- Cocos2d-x中的界面设计



本节内容

- Chapter 6

- 游戏界面设计实例----贪食豆

- UI在Cocos2d-x中的应用

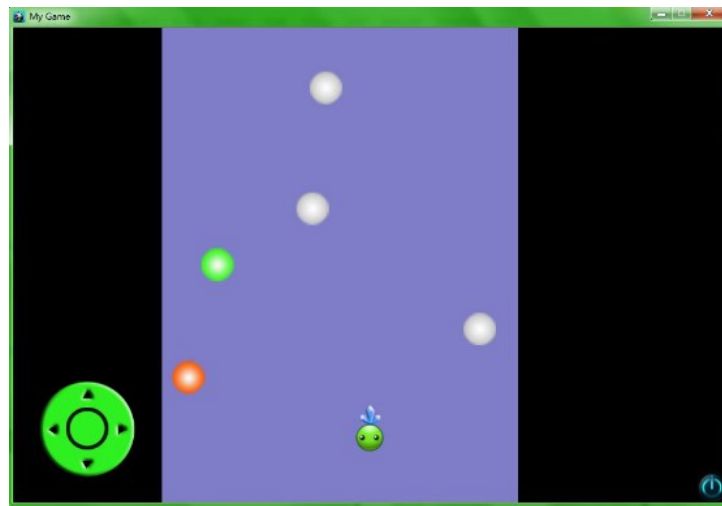
- Cocos2d-x中的场景切换

游戏界面设计实例----贪食豆

- 游戏简介：

本例中实现的是一个简单的通过摇杆控制主角移动的游戏，使用到了自定义的摇杆类来控制一个小豆子“吃到”彩色小球。

游戏操作简单，只要使用摇杆的方向来控制小豆子“吃到”小球即可。



游戏界面设计实例----贪食豆

转场前

游戏界面设计实例----贪食豆

- 游戏设计&实现步骤:

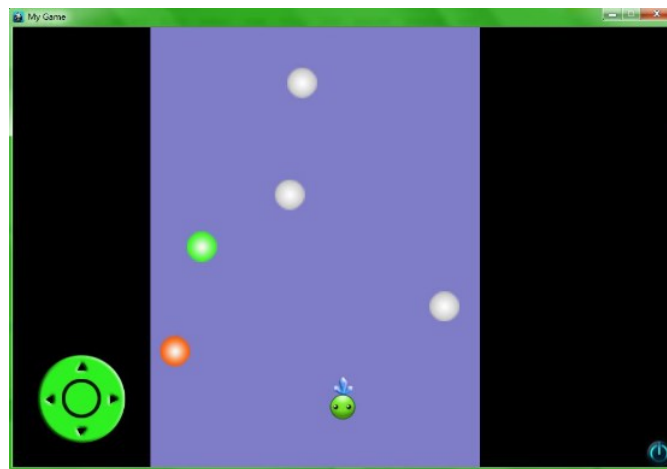
- 解读游戏规则

- UI界面设计、触摸事件、碰撞检测

- 封装摇杆类Joystick

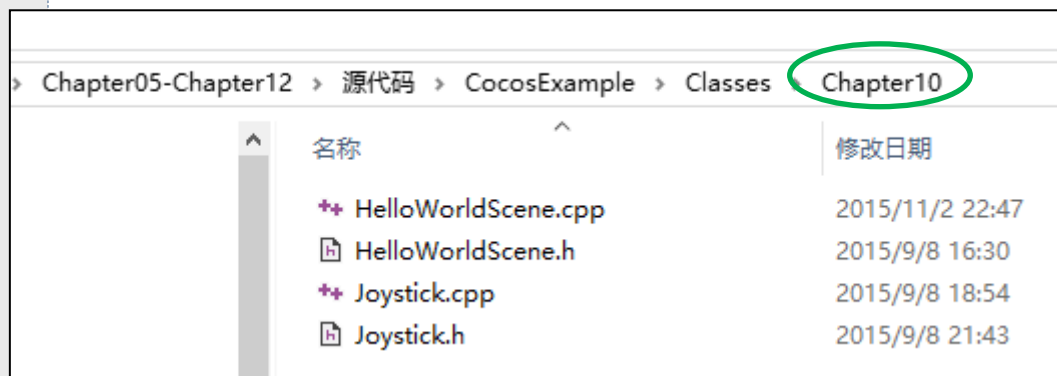
- 在场景中使用摇杆类

- 初始化、update



游戏界面设计实例—贪食豆

• 封装摇杆类Joystick



Chapter05-Chapter12 > 源代码 > CocosExample > Resources > Chapter10



ball.png



ball1.png



ball2.png



bean.png



j-bg.png



j-bg1.png



j-btn.png



ui.png

游戏界面设计实例—贪食豆

- 封装摇杆类Joystick

- 该类继承自Layer类

接收用户交互响应事件

```
ate.cpp  main.cpp  HelloWorldScene.h  Joystick.h  (全局)
22      class Joystick : public Layer
23      {
24      public:
25          Joystick();
26          ~Joystick();
--
```

- 对用户触摸进行响应

```
ate.cpp  main.cpp  HelloWorldScene.h  Joystick.h  HelloWorldScene.cpp  Joystick.cpp  (全局范围)
43      virtual bool onTouchBegan(Touch *pTouch, Event *pEvent);
44      virtual void onTouchMoved(Touch *pTouch, Event *pEvent);
45      virtual void onTouchEnded(Touch *pTouch, Event *pEvent);
46
```


游戏界面设计实例—贪食豆

- 封装摇杆类Joystick

声明摇杆中心、当前位置、半径、控制点

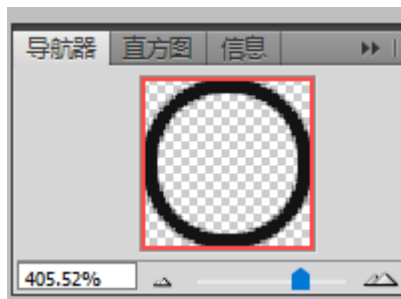
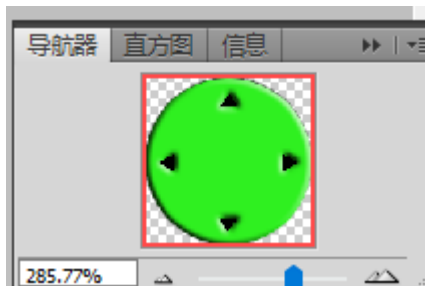
```
37     private:
38         Vec2 m_centerPoint;           // 摇杆中心
39         Vec2 m_currentPoint;          // 摇杆当前位置
40         float m_radius;                // 摇杆半径
41         Sprite* m_jsSprite;           // 摇杆控制点
```



j-bg.png



j-btn.png



游戏界面设计实例—贪食豆

- 封装摇杆类Joystick

声明获取摇杆力度、获取摇杆方向等函数



```
rdScene.h    HelloWorldScene.cpp    Joystick.h  X  Joystick.cpp
文件  (全局范围)
25          // 获取摇杆方向
26          Joystick_dir getDirection();
27
28  private:
29          float getVelocity();    // 摇杆力度
```

摇杆方向用枚举类型表示

```
9  enum Joystick_dir
10  {
11      _LEFT,
12      _RIGHT,
13      _STOP
14  };
```

游戏界面设计实例—贪食豆

- 封装摇杆类Joystick

create()函数和init()函数的声明

Public:

```
17     static HelloWorld* create()  
18     {  
19         HelloWorld *pRet = new HelloWorld();  
20         if (pRet && pRet->init())  
21         {
```

```
27  
28     // 创建摇杆, aPoint是摇杆中心 aRadius是摇杆半径 aJsSprite是摇杆控制点 aJsBg是摇杆背景  
29     static Joystick* create(Vec2 aPoint, float aRadius, char* aJsSprite, char* aJsBg);
```

Private:

摇杆更新函数的声明

```
40     void update(float dt);  
41     // 初始化, aPoint是摇杆中心 aRadius是摇杆半径 aJsSprite是摇杆控制点 aJsBg是摇杆背景  
42     virtual bool init(Vec2 aPoint, float aRadius, char* aJsSprite, char* aJsBg);
```

游戏界面设计实例—贪食豆

- 在JoyStick.cpp文件中实现Joystick类中的功能

Joystick::create()：创建并返回Joystick对象，供其它类调用

```
gate.cpp  main.cpp  HelloWorldScene.h  Joystick.h*  HelloWorldScene.cpp  Joystick.cpp  ~Joystick()
- -> Joystick
42  Joystick* Joystick::create(Vec2 aPoint, float aRadius, char* aJsSprite, char* aJsBg)
43  {
44      Joystick *pRet = new(std::nothrow) Joystick();
45      if (pRet && pRet->init(aPoint, aRadius, aJsSprite, aJsBg)) {
46          pRet->autorelease();
47          return pRet;
48      }
49      else {
50          delete pRet;
51          pRet = NULL;
52          return NULL;
53      }
54  }
```

AutoreleasePool
自动释放池

aPoint是摇杆中心 aRadius是摇杆半径 aJsSprite是摇杆控制点 aJsBg是摇杆背景


游戏界面设计实例—贪食豆

- 在JoyStick.cpp文件中实现Joystick类中的功能

Joystick::init()：初始化对象变量，如设置摇杆半径、摇杆初始位置

```
egate.cpp  main.cpp  HelloWorldScene.h  Joystick.h*  HelloWorldScene.cpp  Joystick.cpp  ↗ ×
→ Joystick  init(Vec2 aPoint, float aRadius,

12  // 初始化 aPoint是摇杆中心 aRadius是摇杆半径 aJsSprite是摇杆控制点 aJsBg是摇杆背景
13  bool Joystick::init(Vec2 aPoint , float aRadius , char* aJsSprite, char* aJsBg)
14  {
15      if (!Layer::init())
16      {
17          return false;
18      }
19      //摇杆初始化
20      m_radius = aRadius;//摇杆半径
21      m_centerPoint = aPoint;//摇杆控制中心
22      m_currentPoint = m_centerPoint; //摇杆当前位置
```




游戏界面设计实例—贪食豆

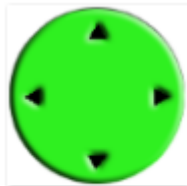
- 在JoyStick.cpp文件中实现Joystick类中的功能

Joystick::init()：初始化对象变量，如设置摇杆图标

```
gate.cpp  main.cpp  HelloWorldScene.h  Joystick.h*  HelloWorldScene.cpp  Joystick.cpp  ↗ ✕  
- → Joystick  
24      m_jsSprite = Sprite::create(aJsSprite); //摇杆中心点贴图  
25      m_jsSprite->setPosition(m_centerPoint);  
26  
27      auto _aJsBg = Sprite::create(aJsBg); //摇杆背景贴图  
28      _aJsBg->setPosition(m_centerPoint);  
29      _aJsBg->setTag(88);  
30      this->addChild(_aJsBg);  
31      this->addChild(m_jsSprite);
```



j-btn.png

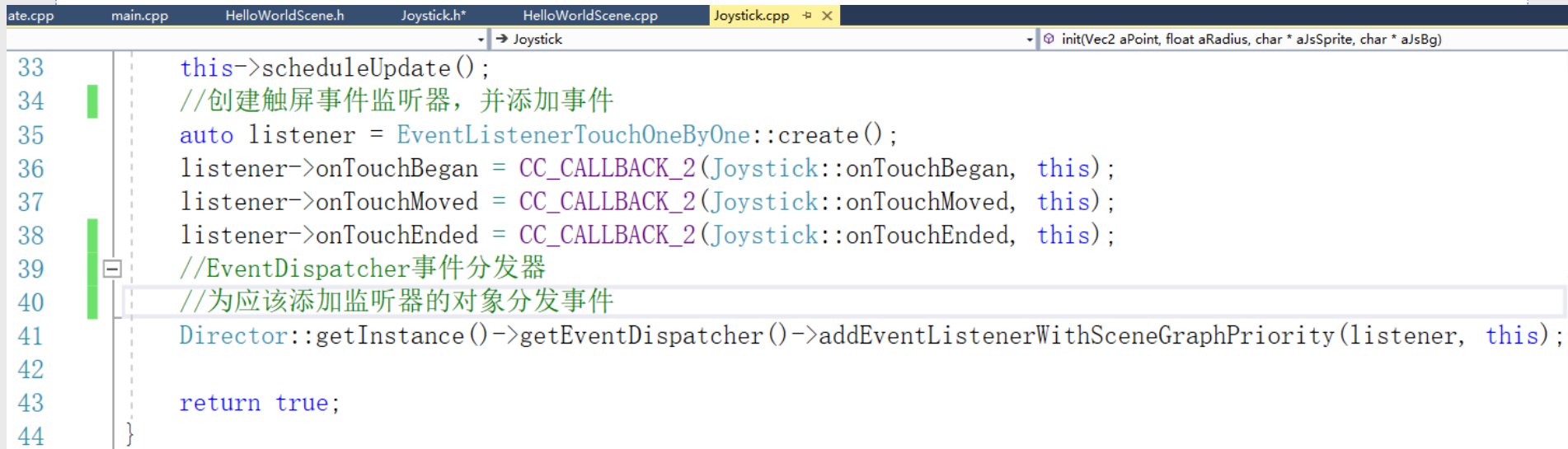


j-bg.png

游戏界面设计实例—贪食豆

- 在JoyStick.cpp文件中实现Joystick类中的功能

Joystick::init()：初始化对象变量，如调用**定时调度函数**来调用update()函数，实现每一帧的刷新；以及添加**监听器**



```
ate.cpp  main.cpp  HelloWorldScene.h  Joystick.h*  HelloWorldScene.cpp  Joystick.cpp  x
→ Joystick  init(Vec2 aPoint, float aRadius, char * aJSprite, char * aJSBg)

33  this->scheduleUpdate();
34  //创建触屏事件监听器，并添加事件
35  auto listener = EventListenerTouchOneByOne::create();
36  listener->onTouchBegan = CC_CALLBACK_2(Joystick::onTouchBegan, this);
37  listener->onTouchMoved = CC_CALLBACK_2(Joystick::onTouchMoved, this);
38  listener->onTouchEnded = CC_CALLBACK_2(Joystick::onTouchEnded, this);
39  //EventDispatcher事件分发器
40  //为应该添加监听器的对象分发事件
41  Director::getInstance()->getEventDispatcher()->addEventListenerWithSceneGraphPriority(listener, this);
42
43  return true;
44 }
```

游戏界面设计实例—贪食豆

- 在JoyStick.cpp文件中实现Joystick类中的功能



监听器Joystick::onTouchBegan():

检测触碰位置是否在摇杆半径内；如是，更新摇杆当前位置

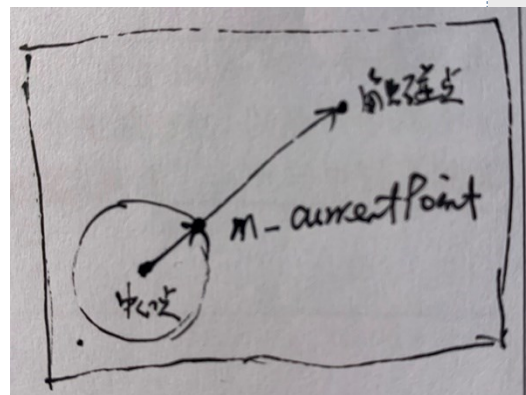
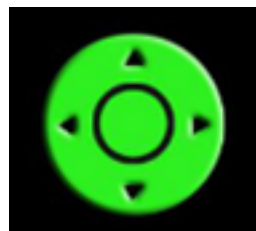
```
ate.cpp  main.cpp  HelloWorldScene.h  Joystick.h*  HelloWorldScene.cpp  Joystick.cpp  ↗ ×
└─→ Joystick
98  bool Joystick::onTouchBegan(Touch *pTouch, Event *pEvent)
99  {
100      auto touchPoint = pTouch->getLocation();
101      if (touchPoint.getDistance(m_centerPoint) > m_radius) {
102          return false;
103      }
104      m_currentPoint = touchPoint; ✓
105      return true;
106  }
```


游戏界面设计实例—贪食豆

- 在JoyStick.cpp文件中实现Joystick类中的功能

监听器Joystick::onTouchMoved():

更新摇杆当前位置，分两种情况：



```
main.cpp  HelloWorldScene.h  Joystick.h*  HelloWorldScene.cpp  Joystick.cpp  x
e
-> Joystick
init(Vec2 aPoint, float aRadius, char * aJsSprite, char * aJsBg)

108 void Joystick::onTouchMoved(Touch *pTouch, Event *pEvent)
109 {
110     auto touchPoint = pTouch->getLocation();
111     if (touchPoint.getDistance(m_centerPoint) > m_radius)
112     {
113         m_currentPoint = m_centerPoint + (touchPoint - m_centerPoint).getNormalized() * m_radius;
114     } else {
115         m_currentPoint = touchPoint;
116     }
117 }
```

游戏界面设计实例—贪食豆

- 在JoyStick.cpp文件中实现Joystick类中的功能

监听器Joystick::onTouchEnded():

触碰结束时，摇杆返回至中心位置



```
gate.cpp  main.cpp  HelloWorldScene.h  Joystick.h*  HelloWorldScene.cpp  Joystick.cpp  X
→ Joystick
119  void Joystick::onTouchEnded(Touch *pTouch, Event *pEvent)
120  {
121      m_currentPoint = m_centerPoint;
122  }
```

游戏界面设计实例—贪食豆

- 在JoyStick.cpp文件中实现Joystick类中的功能



Joystick::getDirection () :

获取摇杆方向

```
56      // 获取摇杆当前方向
57      Joystick_dir Joystick::getDirection()
58      {
59          if ((m_currentPoint - m_centerPoint).x > 0)
60          {
61              return Joystick_dir::_RIGHT;
62          } if ((m_currentPoint - m_centerPoint).x < 0)
63          {
64              return Joystick_dir::_LEFT;
65          }
66          return Joystick_dir::_STOP;
67      }
```

游戏界面设计实例—贪食豆

- 在JoyStick.cpp文件中实现Joystick类中的功能



Joystick::getVelocity():

获取摇杆**力度**



```
gate.cpp  main.cpp  HelloWorldScene.h  Joystick.h*  HelloWorldScene.cpp  Joystick.cpp  X
Joystick

86      // 获取摇杆力度
87      float Joystick::getVelocity()
88      {
89          return m_centerPoint.getDistance(m_currentPoint);
90      }
91
```



游戏界面设计实例—贪食豆

- 创建游戏场景：成员函数声明



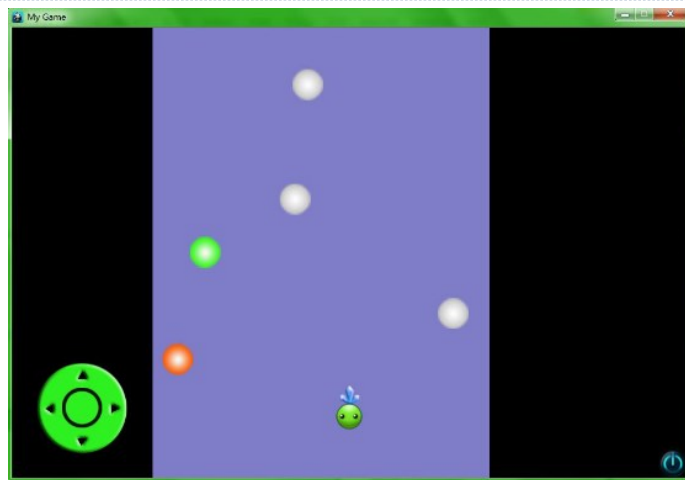
ball.png



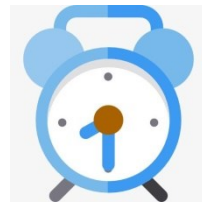
ball1.png



ball2.png



```
16 // a selector callback
17 void menuCloseCallback(cocos2d::Ref* pSender);
18 void addBall1(float dt);
19 void addBall2(float dt);
20 void addBall3(float dt);
21 void removeBall(Sprite* ball);
22
23 void update(float dt);
```



游戏界面设计实例—贪食豆

- 创建游戏场景：成员变量声明

```
26 private:  
27     Sprite* bean;  
28     Size visibleSize;  
29     Joystick* m_joystick;  
30     Vector<Sprite*> ballVector;
```



ball.png



ball1.png

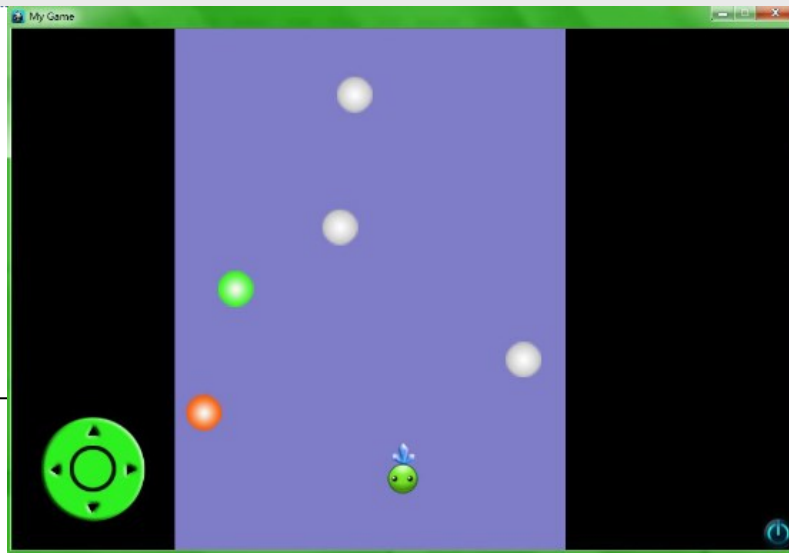


ball2.png

游戏界面设计实例—贪食豆

- 游戏场景初始化：添加颜色背景

```
20 bool HelloWorld::init()  
21 {  
22     if ( !Layer::init() )  
23     {  
24         return false;  
25     }  
26  
27     visibleSize = Director::getInstance()->getVisibleSize();  
28     Vec2 origin = Director::getInstance()->getVisibleOrigin();  
29  
30     //添加颜色层  
31     auto colorLayer = LayerColor::create(Color4B(128, 125, 200, 255), 480, visibleSize.height);  
32     colorLayer->setPosition(Vec2(200, 0));  
33     this->addChild(colorLayer);
```



游戏界面设计实例—贪食豆

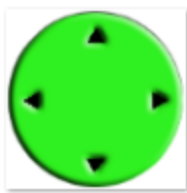
- 游戏场景初始化：添加精灵、摇杆，启动调度器



bean.png



j-btn.png



j-bg.png



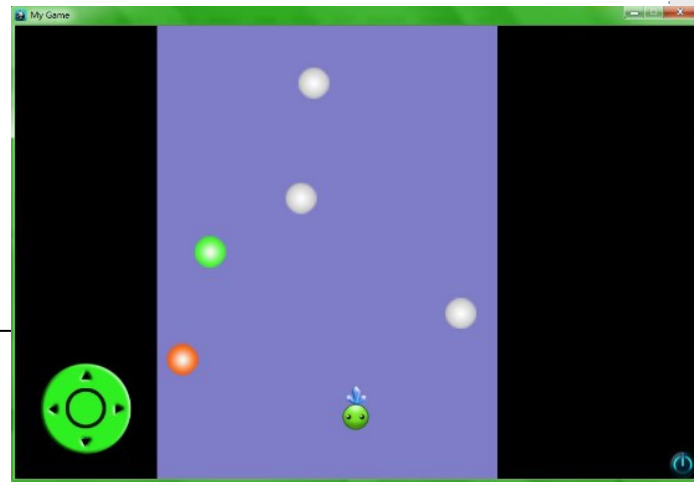
ball.png



ball1.png



ball2.png



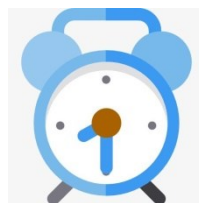
```
//移动精灵
```

```
bean = Sprite::create("Chapter10/bean.png");  
bean->setPosition(Point(visibleSize.width/2, 100));  
this->addChild(bean, 6);
```

```
// 摇杆
```

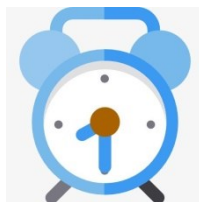
```
m_joystick = Joystick::create(Vec2(100, 100), 50.0f, "Chapter10/j-btn.png", "Chapter10/j-bg.png");  
this->addChild(m_joystick, 4);
```

```
this->schedule(schedule_selector(HelloWorld::addBall1), 1.0f);  
this->schedule(schedule_selector(HelloWorld::addBall2), 2.0f);  
this->schedule(schedule_selector(HelloWorld::addBall3), 3.0f);  
this->scheduleUpdate();
```



游戏界面设计实例—贪食豆

• 游戏场景初始化：调度器



ball.png



ball1.png



ball2.png

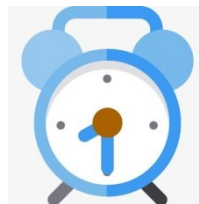
```
60      | |      this->schedule(schedule_selector(HelloWorld::addBall3), 3.0f);  
61      | |      this->scheduleUpdate();
```

Cocos2d-x调度器为游戏提供定时事件和定时调用服务。所有Node对象都知道如何调度和取消调度事件，使用调度器有几个好处：

1. 每当Node不再可见或已从场景中移除时，调度器会停止。
2. Cocos2d-x暂停时，调度器也会停止。当Cocos2d-x重新开始时，调度器也会自动继续启动。
3. Cocos2d-x封装了一个供各种不同平台使用的调度器，使用此调度器你不用关心和跟踪你所设定的定时对象的销毁和停止，以及崩溃的风险。

游戏界面设计实例—贪食豆

- 游戏场景初始化：调度器



```
60      |      this->schedule(schedule_selector(HelloWorld::addBall3), 3.0f);  
61      |      this->scheduleUpdate();
```

Cocos2d-x中有三种定时器：schedule, scheduleUpdate, scheduleOnce。了解了其功能你便会发现定时器真是很简单，很方便，下面是它们的异同：

1. scheduleUpdate(); 此函数是Node的成员函数，每个Node只要调用scheduleUpdate()，那么这个Node就会自动刷新当前类的update(float dt)函数体。scheduleUpdate()默认每一帧都会调用update函数。

游戏界面设计实例—贪食豆

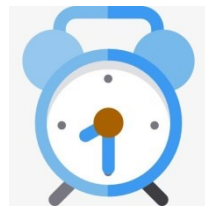
```
60      | |      this->schedule(schedule_selector(HelloWorld::addBall3), 3.0f);  
61      | |      this->scheduleUpdate();
```

2. schedule的作用与scheduleUpdate()函数相似，但是scheduleUpdate()默认每一帧都会调用update函数，而schedule则可以自定义刷新的函数体和时间间隔。

* [1]schedule(selector); 参数：目标函数，即自定义的更新函数。该函数等同于scheduleUpdate，默认每一帧都调用目标函数。

* [2]schedule(selector,interval); 参数：目标函数，更新时间。

* [3]schedule(selector,interval,repeat,delay); 参数：目标函数，更新时间，更新次数，每次等待时间。



游戏界面设计实例—贪食豆

```
60 | | this->schedule(schedule_selector(HelloWorld::addBall3), 3.0f);  
61 | | this->scheduleUpdate();
```

3. `scheduleOnce(selector, delay)`; 参数：目标函数，等待时间。只执行一次，可以指定刷新的函数体。

停用定时器的方法：

2. 停止自定义更新函数。

3. 停止所有更新函数。

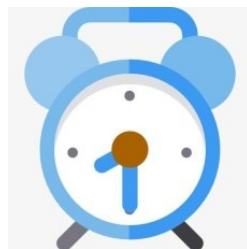
1. 停止默认的update更新函数。

2 `unschedule(selector)`;

3 `unscheduleAllSelectors()`

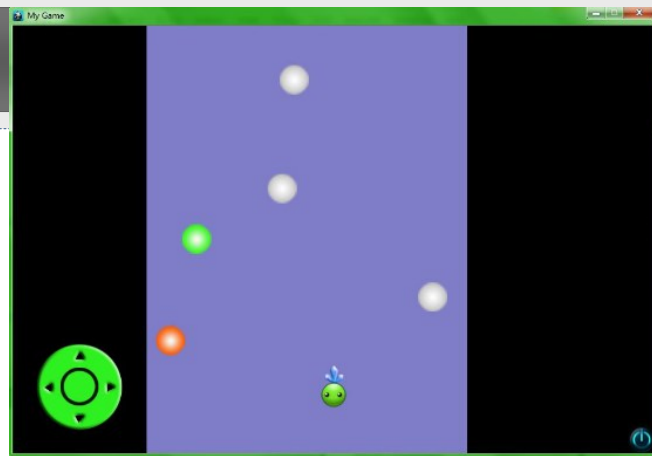
1 `unscheduleUpdate()`;

参数：自定义的更新函数。



游戏界面设计实例—贪食豆

- addBall()函数:
 - 创建球贴图精灵, 生成随机位置 ?
 - 创建移动轨迹



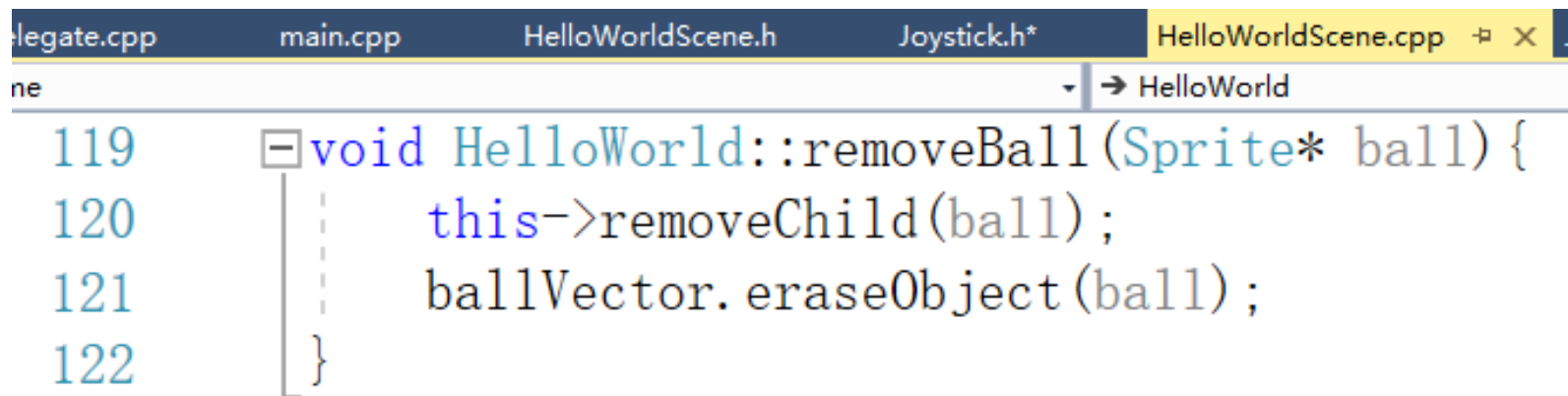
gate.cpp main.cpp HelloWorldScene.h Joystick.h* HelloWorldScene.cpp x Joystick.cpp 200 480 addBall1(float dt)

```
75 void HelloWorld::addBall1(float dt) {
76     auto ball1 = Sprite::create("Chapter10/ball.png");//使用图片创建小球
77     //设置小球的初始位置, 这里在x方向使用了随机函数rand使得小球在随机位置出现
78     ball1->setPosition(Point(207 + rand() % 460, visibleSize.height));
79     ball1->setTag(1);
80     this->addChild(ball1, 5);
81     this->ballVector.pushBack(ball1);//将小球放进Vector数组
82     auto moveTo = MoveTo::create(rand() % 5, Point(ball1->getPositionX(), -10));//移动动作
83     //当小球移动到屏幕下方时回调removeBall函数, 移除小球
84     auto actionDone = CallFunc::create(CC_CALLBACK_0(HelloWorld::removeBall, this, ball1));
85     auto sequence = Sequence::create(moveTo, actionDone, nullptr);
86     ball1->runAction(sequence);//执行动作
87 }
```

游戏界面设计实例—贪食豆

- removeBall()函数:

- 自动回收小球; 此功能只会在两种情况下调用。
- 分别是MoveTo完成后,
- 以及小球被玩家碰撞 (稍后在update中实现)。



```
delegate.cpp  main.cpp  HelloWorldScene.h  Joystick.h*  HelloWorldScene.cpp  [X]
ne  [v] → HelloWorld

119  void HelloWorld::removeBall(Sprite* ball) {
120      this->removeChild(ball);
121      ballVector.eraseObject(ball);
122  }
```

游戏界面设计实例—贪食豆

- update()函数:

- 检测玩家有没有与下落的球发生碰撞；若有，则消去球。
- 检测碰撞最简单的方法就是判断两个模型有无发生重合

```
gate.cpp  main.cpp  HelloWorldScene.h  Joystick.h*  HelloWorldScene.cpp  Joystick.cpp
HelloWorld
update(float dt)
126 void HelloWorld::update(float dt) {
127     //遍历检测碰撞
128     for (auto ball:ballVector)
129     {
130         //获取bean的边界并进行矩形碰撞检测
131         if (bean->getBoundingBox().intersectsRect(ball->getBoundingBox()))
132         {
133             auto actionDown = CallFunc::create(CC_CALLBACK_0(HelloWorld::removeBall, this, ball));
134             ball->runAction(actionDown);

```

游戏界面设计实例—贪食豆

- **update()函数:**
 - 根据摇杆方向令小豆子精灵发生位移

```
111 // 控制角色移动
112 if (m_joystick->getDirection() == Joystick_dir::_RIGHT
113     &&bean->getPositionX()+bean->getContentSize().width/2<=680)
114 {
115     bean->setPositionX(bean->getPositionX()+4);
116 }
117 if (m_joystick->getDirection() == Joystick_dir::_LEFT
118     &&bean->getPositionX()-bean->getContentSize().width/2>=200)
119 {
120     bean->setPositionX(bean->getPositionX() - 4);
121 }
122 if (m_joystick->getDirection() == Joystick_dir::_STOP)
123 {
124     bean->setPositionX(bean->getPositionX());
125 }
```


游戏界面设计实例—贪食豆

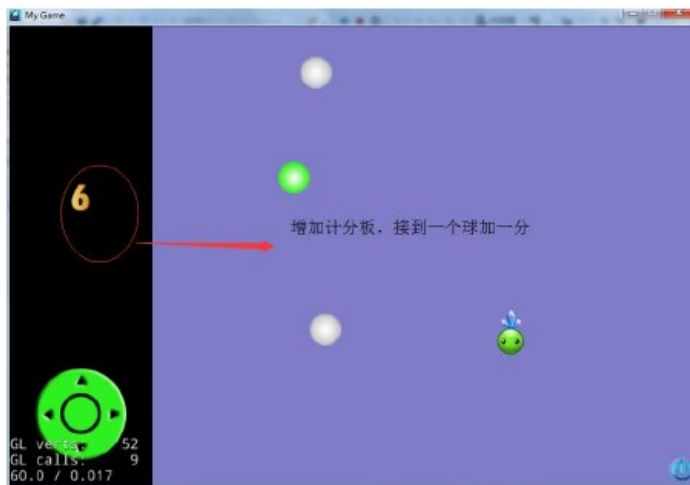
• 游戏优化方向？



增加上下移动

增加移动边界检测

- 摇杆半径大小
- 摇杆速度与精灵速度

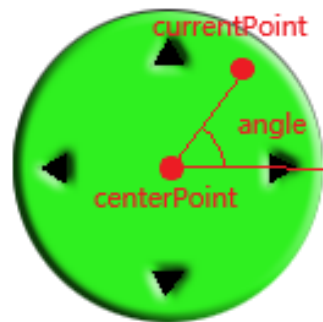


增加计分板。接到 1 个球加 1 分

计分板还可以根据 接到不同颜色的球 加不同的分数



ball3.png

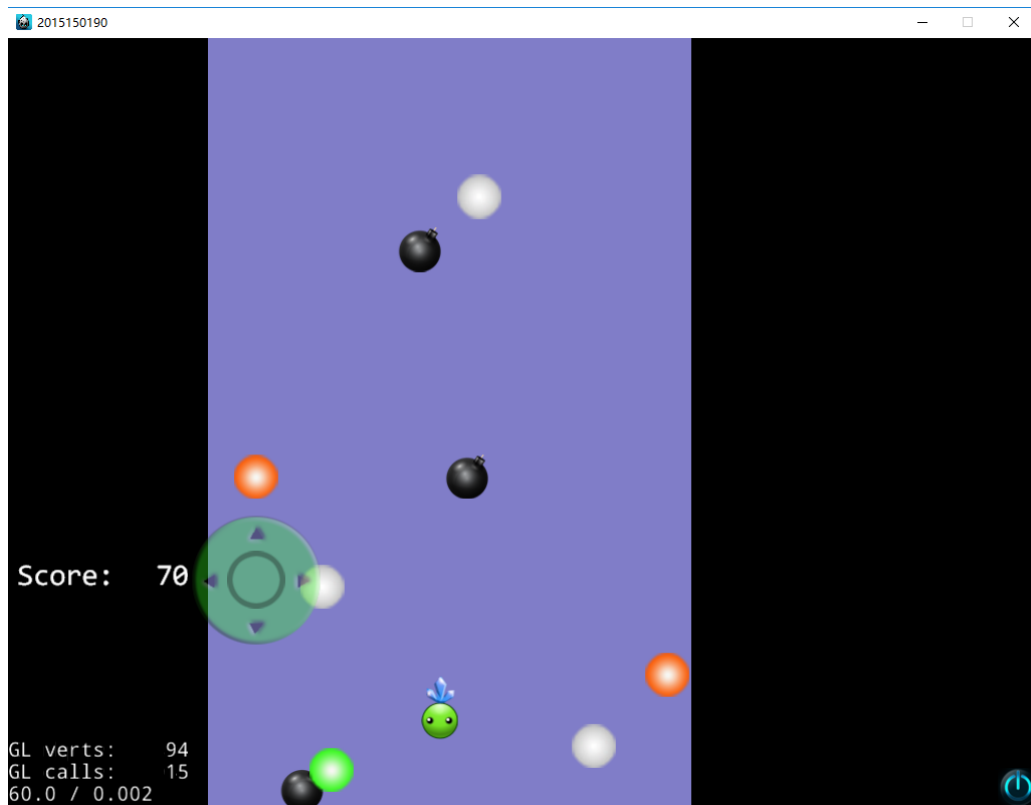


任意方向



游戏界面设计实例—贪食豆

- 游戏优化方向？



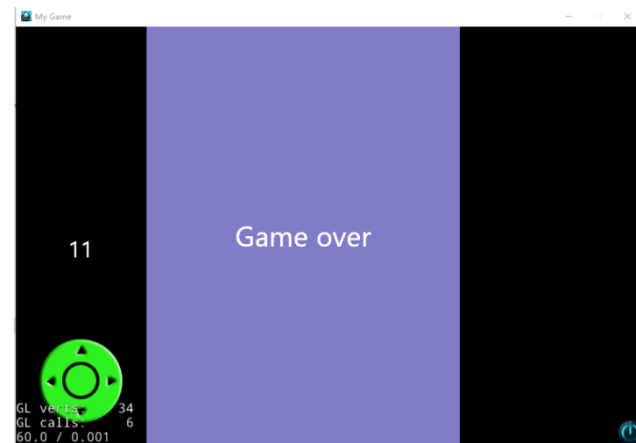
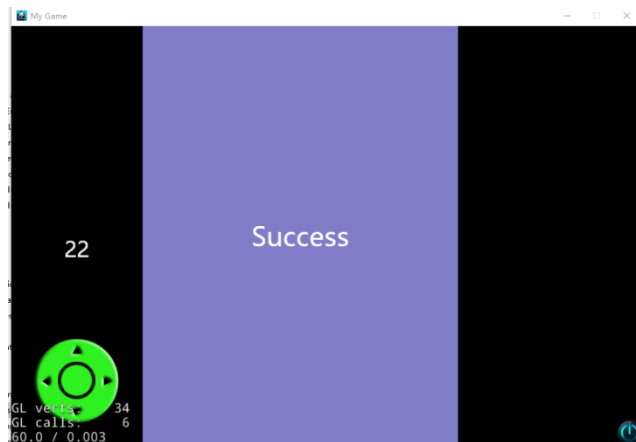
游戏界面设计实例----贪食豆

转场前

游戏界面设计实例—贪食豆

- 游戏优化方向?

- 添加 **Replay** 按钮
- 添加 **初始**界面
- 添加 **Level 2, 3, 4 ...**



本节内容

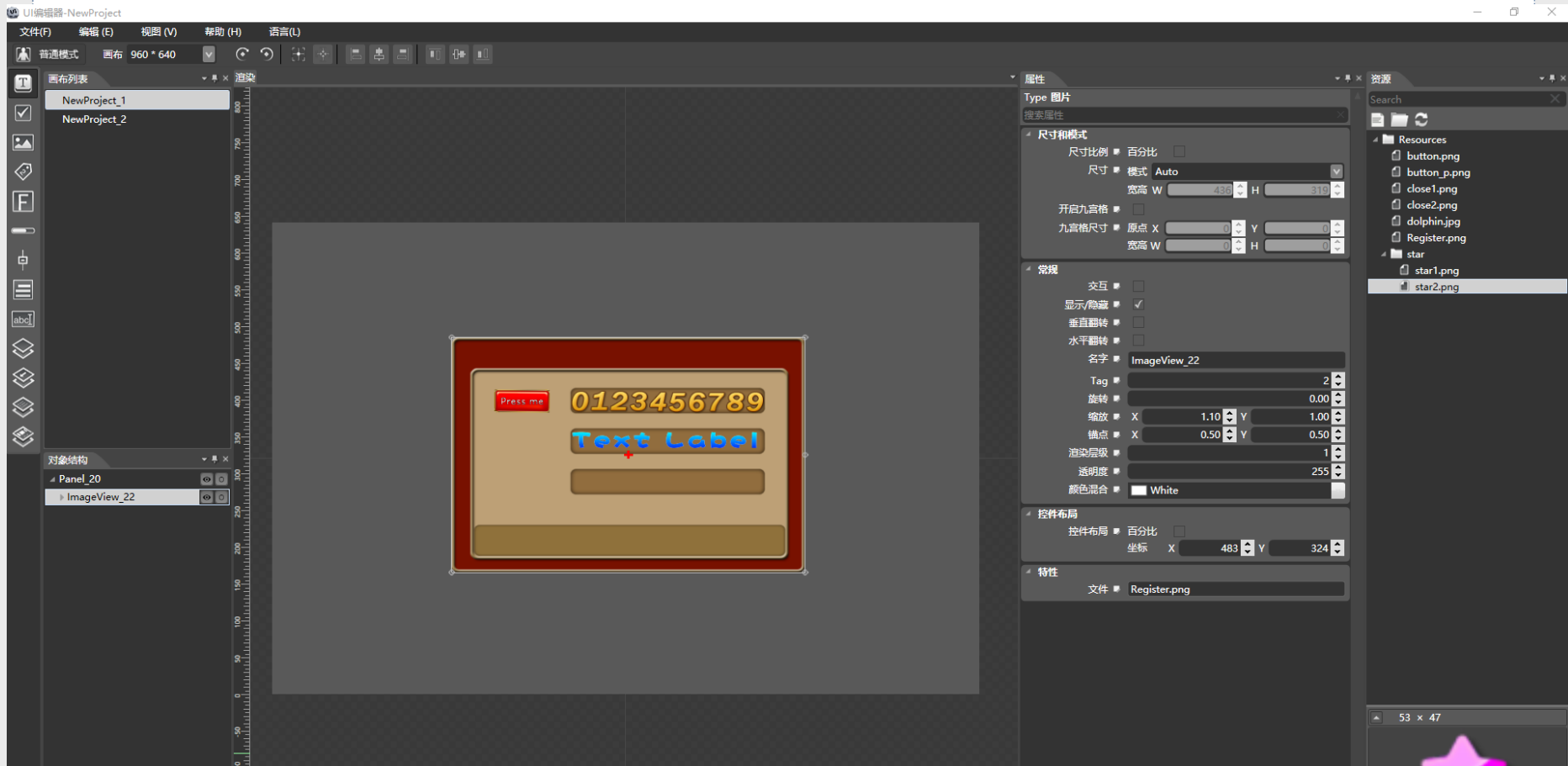
- Chapter 6

- 游戏界面设计实例----贪食豆

- UI在Cocos2d-x中的应用

- Cocos2d-x中的场景切换

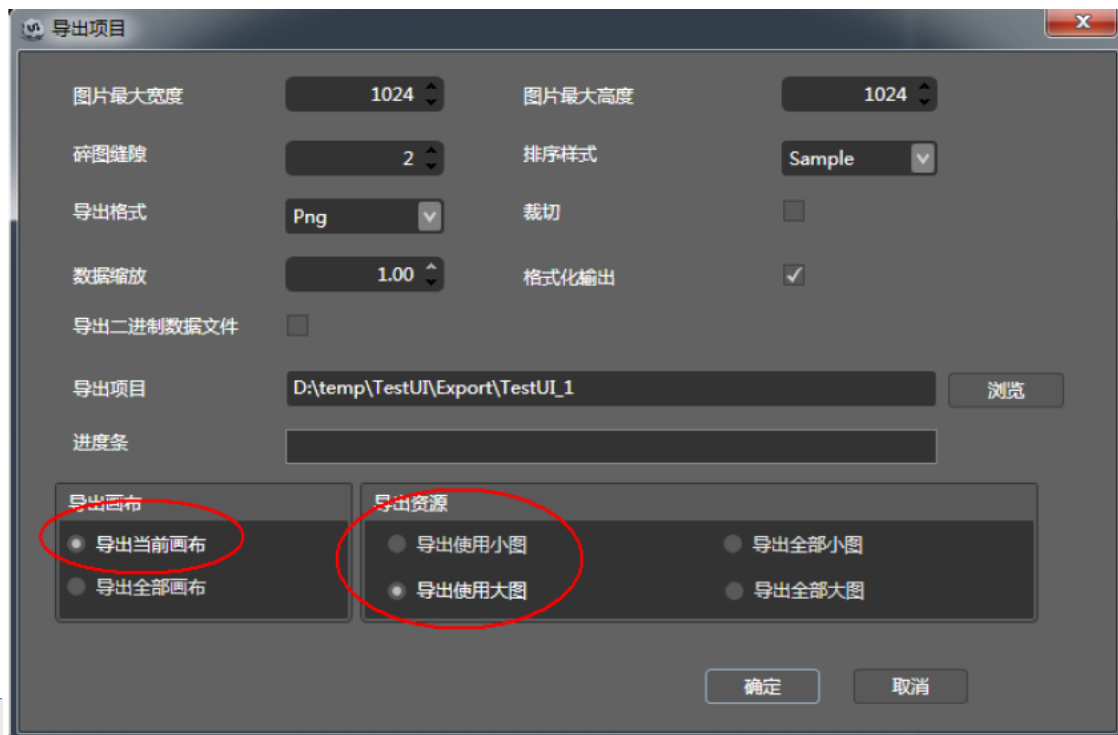
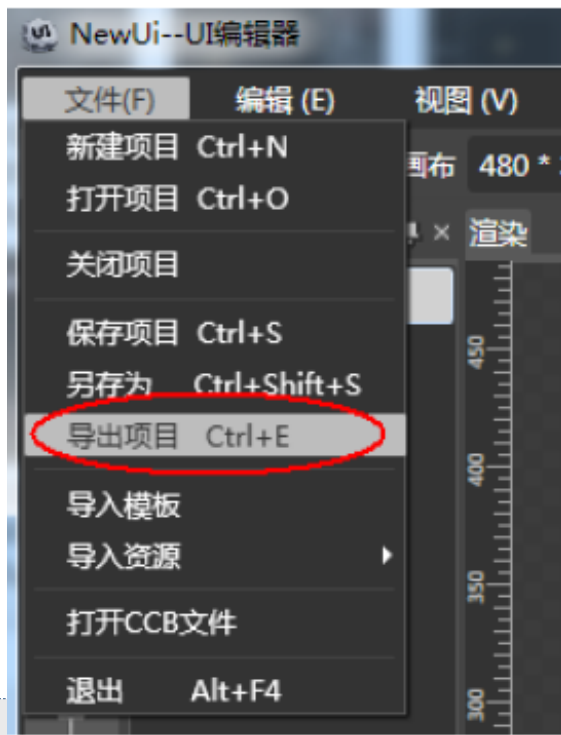
UI在Cocos2d-x中的应用



UI在Cocos2d-x中的应用

导出项目

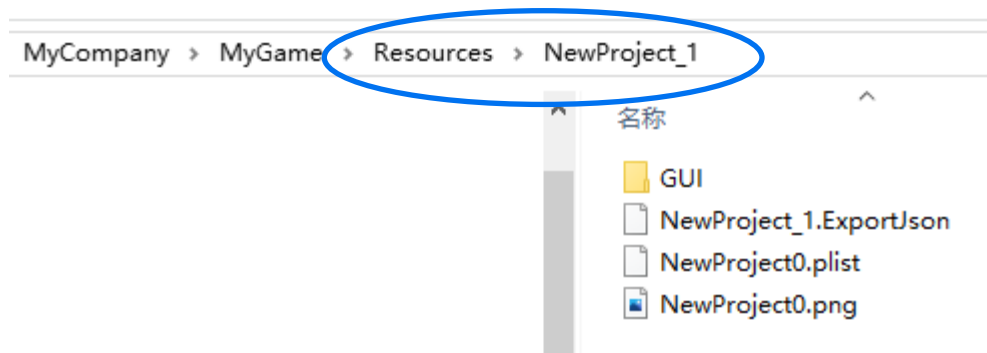
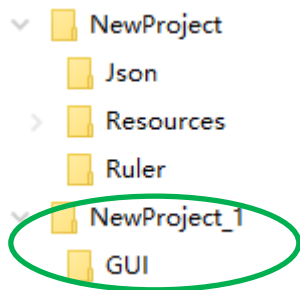
选择文件->导出项目，按默认参数即可导出项目。



UI在Cocos2d-x中的应用

导出项目

导出完毕后，把导出的文件夹拷贝到cocos2d-x project的Resource文件夹下，就能够在项目中使用。



UI在Cocos2d-x中的应用

调用UI场景

最后，在需要使用这个UI的场景，添加以下代码

```
auto test_UI = GUIReader::getInstance()->widgetFromJsonFile("NewProject_1/NewProject_1.ExportJson");  
addChild(test_UI);
```



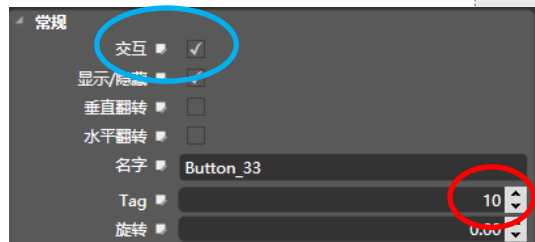
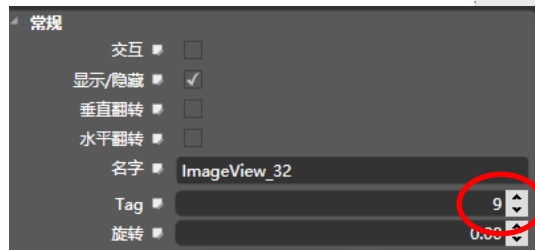
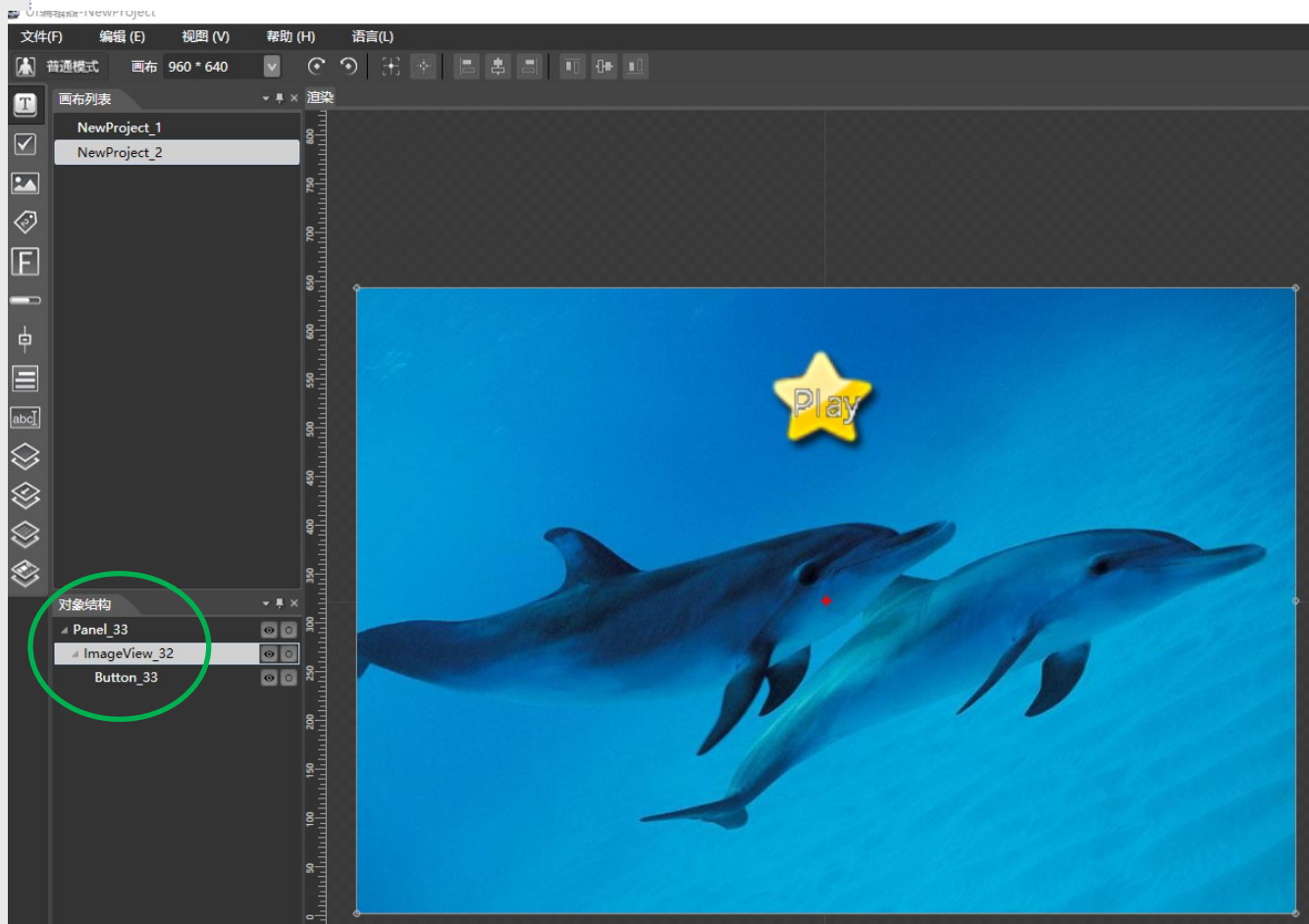
注意：别忘了在文件开头添加

如果要访问此UI中的某个资源，
可以通过tag来操作

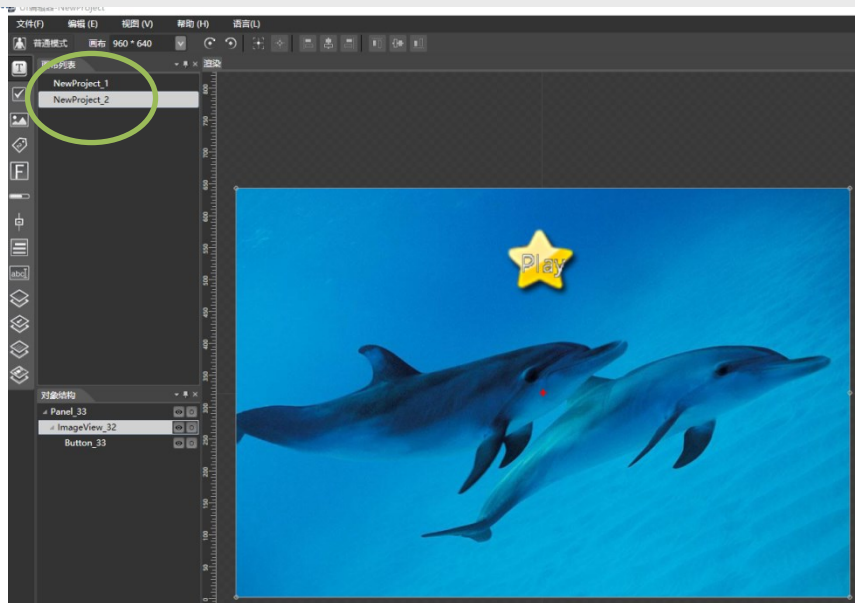
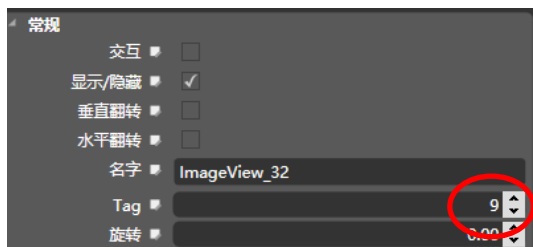
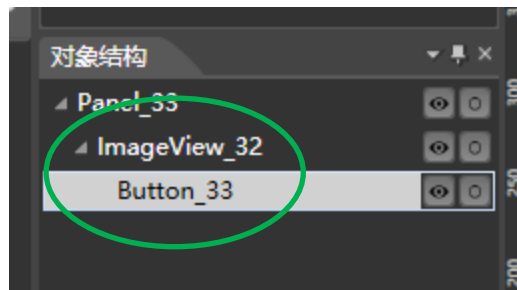
`test_UI->getChildByTag()`

```
1  #include "HelloWorldScene.h"  
2  #include "SimpleAudioEngine.h"  
3  #include "ui/CocosGUI.h"  
4  #include "cocosstudio/CocoStudio.h"  
5  
6  USING_NS_CC;  
7  using namespace ui;  
8  using namespace cocostudio;
```

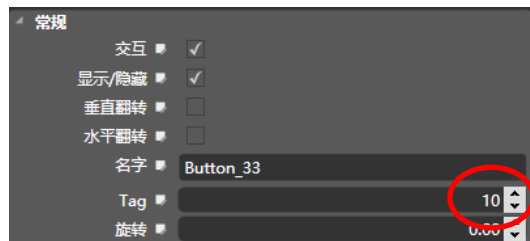
UI在Cocos2d-x中的应用



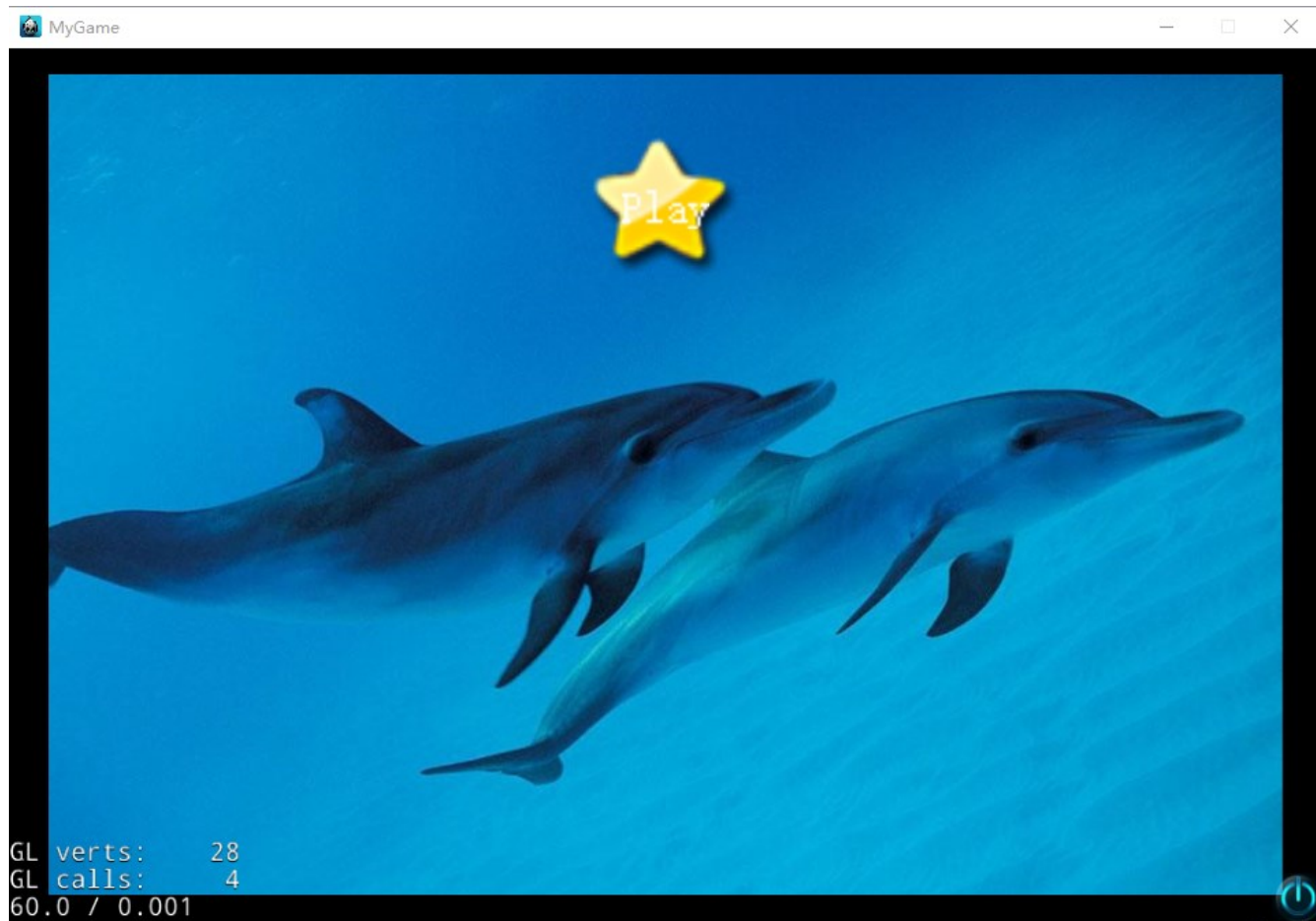
UI在Cocos2d-x中的应用



```
83 auto test_UI = GUIReader::getInstance()->widgetFromFile("NewProject_2/NewProject_2.ExportJson");
84
85 addChild(test_UI);
86
87 auto imageView = (ImageView *) test_UI->getChildByTag(9);
88 // add button click callback
89 auto btn_test_UI = (Button *) imageView->getChildByTag(10);
90 btn_test_UI->addTouchEventListeners(CC_CALLBACK_2(HelloWorld::onClick, this));
```



UI在Cocos2d-x中的应用



UI在Cocos2d-x中的应用

转场前

本节内容

- Chapter 6

- 游戏界面设计实例----贪食豆
- UI在Cocos2d-x中的应用
- Cocos2d-x中的场景切换

Cocos2d-x中的场景切换

场景切换的方式

有很多场景切换的方式，每种都有特定的方法，让我们来看看：

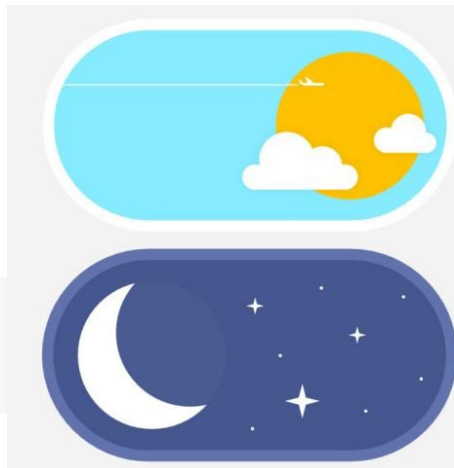
```
auto myScene = Scene::create();
```

runWithScene() 用于开始游戏，加载第一个场景。只用于第一个场景！

```
Director::getInstance()->runWithScene(myScene);
```

replaceScene() 使用传入的场景替换当前场景来切换画面，当前场景被释放。这是切换场景时最常用的方法。

```
Director::getInstance()->replaceScene(myScene);
```



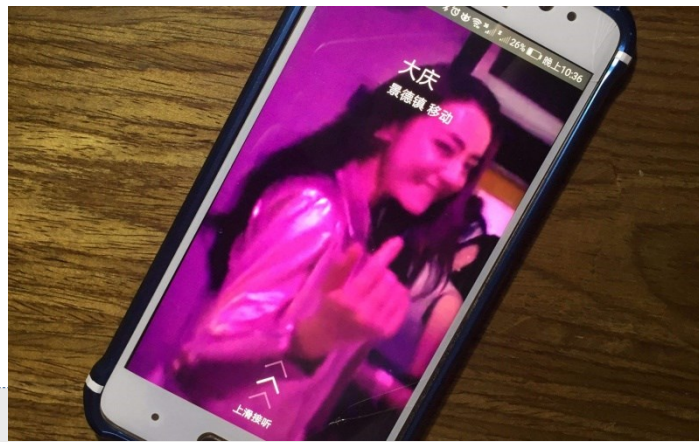
Cocos2d-x中的场景切换

pushScene() 将当前运行中的场景暂停并压入到场景栈中，再将传入的场景设置为当前运行场景。只有存在正在运行的场景时才能调用该方法。

```
Director::getInstance()->pushScene(myScene);
```

popScene() 释放当前场景，再从场景栈中弹出栈顶的场景，并将其设置为当前运行场景。如果栈为空，直接结束应用。

```
Director::getInstance()->popScene();
```



Cocos2d-x中的场景切换

场景切换的效果设置

```
// Transition Fade
```

```
Director::getInstance()->replaceScene(TransitionFade::create(0.5, myScene, Color3B(0,255,255))));
```

```
// FlipX
```

```
Director::getInstance()->replaceScene(TransitionFlipX::create(2, myScene));
```

```
// Transition Slide In
```

```
Director::getInstance()->replaceScene(TransitionSlideInT::create(1, myScene) );
```



Cocos2d-x中的场景切换

```
90      btn_test_UI->addEventListener(CC_CALLBACK_2(HelloWorld::onClick, this));
```

```
30      void onClick(Ref *pSender, Widget::TouchEventType type);
```

```
149      void HelloWorld::onClick(Ref *pSender, Widget::TouchEventType type)
150      {
151          switch (type)
152          {
153              break;
154              case cocos2d::ui::Widget::TouchEventType::ENDED:
155
156                  Director::getInstance()->replaceScene(MapScene::createScene());
157                  break;
158              default:
159                  break;
160          }
161      }
```

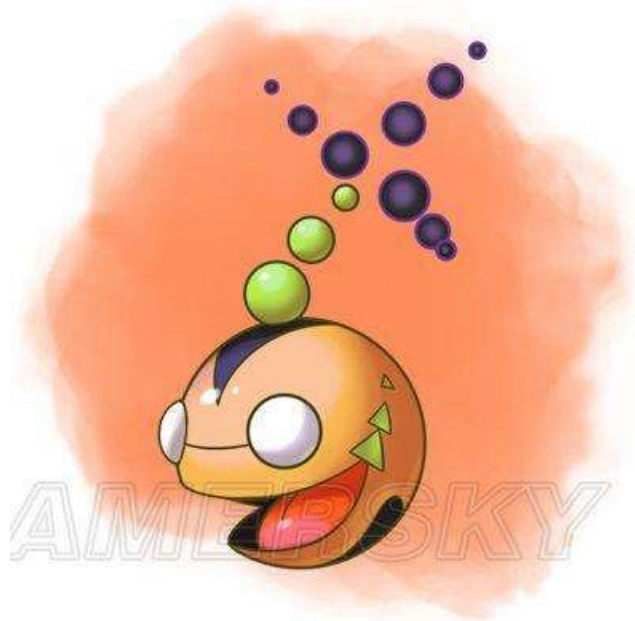
Cocos2d-x中的场景切换

转场前

实验2 游戏交互界面设计

•生成 & 优化 “贪食豆” 游戏

- 顺利运行游戏
- 修改游戏显示名称（学号、姓名）
- 增加摇杆上下移动功能
- 增加计分板功能
- 增加 UI 登录界面（含 Play 按钮）
- 增加 Replay 按钮
- 其它优化功能...



小结

- 游戏界面设计实例----贪食豆
- UI在Cocos2d-x中的应用
- Cocos2d-x中的场景切换