Cocos2dx第15周实验报告

1. 参考资料
2. <http://docs.cocos.com/cocos2d-x/manual/zh/physics/> （cocos官方文档）
3. 课件ppt
4. 实验步骤

**·**基本需求：

1. 控制板子左右移动
2. 在顶部生成小砖块
3. 使用关节固定球与板子
4. 为板子、球、以及砖块设置物理属性
5. 砖、球碰撞则消去砖头，球与地板碰撞则游戏结束

**·**详细实现过程：

1.

增加键盘监听器：

1. auto keyboardListener = EventListenerKeyboard::create();
2. keyboardListener->onKeyPressed = CC\_CALLBACK\_2(HitBrick::onKeyPressed, **this**);
3. keyboardListener->onKeyReleased = CC\_CALLBACK\_2(HitBrick::onKeyReleased, **this**);
4. \_eventDispatcher->addEventListenerWithSceneGraphPriority(keyboardListener, **this**);

实现onKeyPressed和onKeyReleased函数：

onKeyPressed判断当前按钮，并设定相应方向的初始速度，调用Node的方法getPhysicsBody()得到物理属性，设置初始速度（注意方向）：

1. **void** HitBrick::onKeyPressed(EventKeyboard::KeyCode code, Event\* event) {
3. **switch** (code) {
4. **case** cocos2d::EventKeyboard::KeyCode::KEY\_LEFT\_ARROW:
5. **case** cocos2d::EventKeyboard::KeyCode::KEY\_A:
6. currDir = "Left";
7. player->getPhysicsBody()->setVelocity(Vec2(-500, 0));
8. **break**;
9. **case** cocos2d::EventKeyboard::KeyCode::KEY\_RIGHT\_ARROW:
10. **case** cocos2d::EventKeyboard::KeyCode::KEY\_D:
11. currDir = "Right";
12. player->getPhysicsBody()->setVelocity(Vec2(500, 0));
13. **break**;
15. **case** cocos2d::EventKeyboard::KeyCode::KEY\_SPACE: // 开始蓄力
16. spFactor = 0;
17. spHolded = **true**;
18. **break**;
19. **default**:
20. **break**;
21. }
22. }

onKeyReleased函数，当按钮释放之后，player停止移动：

1. // 释放按键
2. **void** HitBrick::onKeyReleased(EventKeyboard::KeyCode code, Event\* event) {
3. **switch** (code) {
4. **case** cocos2d::EventKeyboard::KeyCode::KEY\_LEFT\_ARROW:
5. **case** cocos2d::EventKeyboard::KeyCode::KEY\_RIGHT\_ARROW:
6. **case** cocos2d::EventKeyboard::KeyCode::KEY\_A:
7. **case** cocos2d::EventKeyboard::KeyCode::KEY\_D:
8. // 停止运动
9. // Todo
10. player->getPhysicsBody()->setVelocity(Vec2(0, 0));
11. **break**;
12. **case** cocos2d::EventKeyboard::KeyCode::KEY\_SPACE:   // 蓄力结束，小球发射
13. **if** (onBall) {
14. m\_world->removeAllJoints();
15. ball->getPhysicsBody()->setVelocity(Vec2(0, spFactor));
16. }
17. onBall = **false**;
18. spHolded = **false**;
19. **break**;
20. **default**:
21. **break**;
22. }
23. }

2.

在下面的函数中，设定每个砖块的位置，然后设定每个砖块的物理属性，设置其碰撞的掩码属性。

* categoryBitmask 类别掩码,标明物体的**类别。**
* collisionBitmask 碰撞掩码,标明该物体会和哪种类别的物体发生碰撞。和对方的 categoryBitmask相与不为0，且对方的collisionBitmask和自己的categoryBitmask相与也不为0，则会发生碰撞。
* contactTestBitmask 接触测试掩码,标明该物体对和谁的**碰撞检测**感兴趣，若双方的contactTestBitmask和categoryBitmask相与都不为0，则系统会**传达**碰撞信息。如掩码与为0，则不传达碰撞信息，即使发生了碰撞，系统也不会做出反馈（比如调用监听器）

1. **void** HitBrick::BrickGeneraetd() {
3. **for** (**int** i = 0; i < 3; i++) {
4. **int** cw = 0;
5. **while** (cw <= visibleSize.width) {
6. auto box = Sprite::create("box.png");
7. //box->setAnchorPoint(Vec2(0.5, 0.5));
8. // 为砖块设置刚体属性
9. box->setPosition(cw+box->getContentSize().width/2, visibleSize.height-i\*box->getContentSize().height-box->getContentSize().height/2);
10. auto boxPhysicsBody = PhysicsBody::createBox(Size(box->getContentSize().width, box->getContentSize().height), PhysicsMaterial(50.0f, 1.0, 1.0f));
11. boxPhysicsBody->setCategoryBitmask(0x2);
12. boxPhysicsBody->setCollisionBitmask(0x2);
13. boxPhysicsBody->setContactTestBitmask(0x2);
14. boxPhysicsBody->setDynamic(**false**);
15. box->setTag(BOX);
16. box->setPhysicsBody(boxPhysicsBody);
17. **this**->addChild(box, 2);
18. cw += box->getContentSize().width;
19. }
20. }
21. }

默认情况下，球是放置在板子上。由于二者不相连，未开始游戏当板子移动时，球会脱离板子。所以需要将球固定到板子上跟随板子移动，当游戏开始后，断开连接即可。

1. **void** HitBrick::setJoint() {
2. joint1= PhysicsJointPin::construct(
3. ball->getPhysicsBody(),player->getPhysicsBody(), ball->getPosition()
4. );
5. m\_world->addJoint(joint1);
6. }

利用钉子，将球和板子固定在一起。

1. 为板子、球、以及砖块设置物理属性

碰撞属性的设置，板子的CategoryBitmask、CollisionBitmask、ContactTestBitmask都设置为1；球的CategoryBitmask、CollisionBitmask、ContactTestBitmask都设置为3；砖块的CategoryBitmask、CollisionBitmask、ContactTestBitmask都设置为2。

因为板子的CollisionBitmask和CollisionBitmask与球的CategoryBitmask相与不为1，球的CollisionBitmask和CollisionBitmask与板子的CategoryBitmask相与也不为1，所以二者可以发生接触测试并发生碰撞。

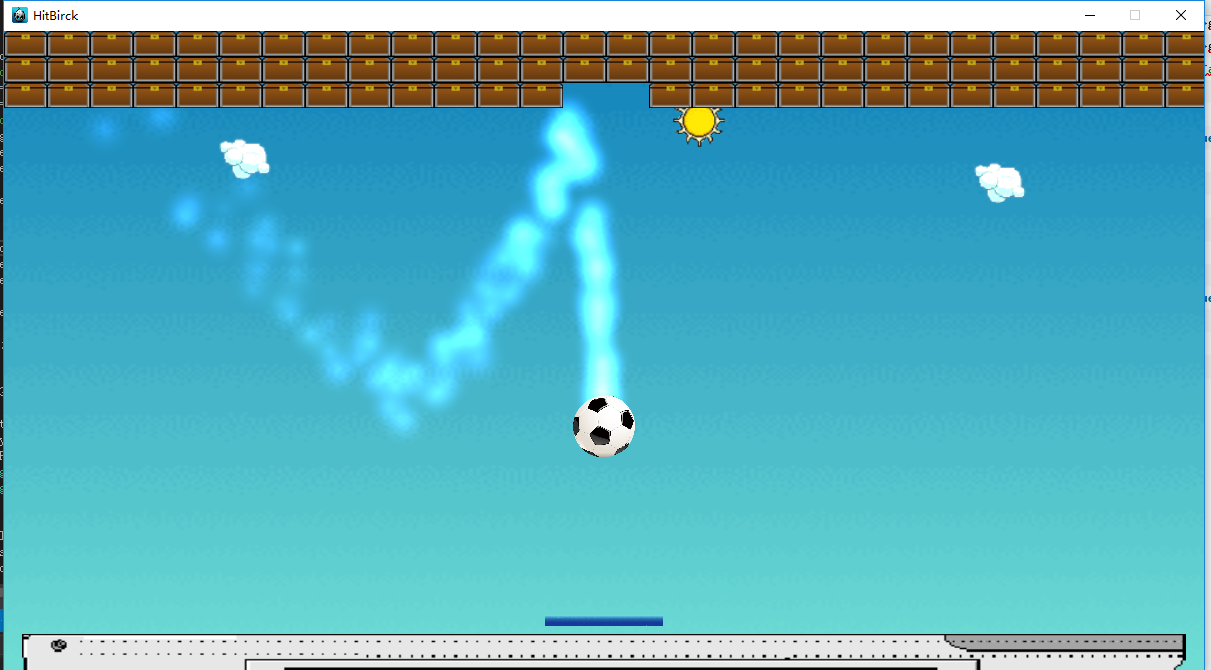
同理，球和砖块也可以可以发生接触测试并发生碰撞。

对板子来说，由于板子始终位于船上方，不受各种力的影响，所以可将其设为静态刚体；另一方面，和球的撞击需要更改球的方向，所以可以增加摩擦系数来改变球的运动；对于球，可以取消其受重力的影响，这样在整个运动的过程中，不会因为重力而使得球无法击中砖块；对于砖块，设置其为静态刚体，不受力的作用。

1. // 设置板的刚体属性
2. auto playerPhysicsBody = PhysicsBody::createBox(Size(player->getContentSize().width, player->getContentSize().height), PhysicsMaterial(100.0f, 1.0, 1.0f));
3. playerPhysicsBody->setCategoryBitmask(0x1);
4. playerPhysicsBody->setCollisionBitmask(0x1);
5. playerPhysicsBody->setContactTestBitmask(0x1);
6. playerPhysicsBody->setDynamic(**false**);
7. player->setTag(PLAYER);
8. player->setPhysicsBody(playerPhysicsBody);
10. // 设置球的刚体属性
11. auto ballPhysicsBody = PhysicsBody::createCircle(ball->getContentSize().width/2.0, PhysicsMaterial(100.0f, 1.0, 1.0f));
12. ballPhysicsBody->setCategoryBitmask(0x3);
13. ballPhysicsBody->setCollisionBitmask(0x3);
14. ballPhysicsBody->setContactTestBitmask(0x3);
15. ballPhysicsBody->setGravityEnable(**false**);
16. ballPhysicsBody->setRotationEnable(false);
17. ball->setTag(BALL);
18. ball->setPhysicsBody(ballPhysicsBody);
20. // 为砖块设置刚体属性
21. box->setPosition(cw+box->getContentSize().width/2, visibleSize.height-i\*box->getContentSize().height-box->getContentSize().height/2);
22. auto boxPhysicsBody = PhysicsBody::createBox(Size(box->getContentSize().width, box->getContentSize().height), PhysicsMaterial(50.0f, 1.0, 1.0f));
23. boxPhysicsBody->setCategoryBitmask(0x2);
24. boxPhysicsBody->setCollisionBitmask(0x2);
25. boxPhysicsBody->setContactTestBitmask(0x2);
26. boxPhysicsBody->setDynamic(**false**);
27. box->setTag(BOX);
28. box->setPhysicsBody(boxPhysicsBody);

碰撞检测，判断球是否与砖块或者船碰撞，利用node节点设置的Tag属性来区分当前碰撞双方：

1. auto nodeA = contact.getShapeA()->getBody()->getNode();
2. auto nodeB = contact.getShapeB()->getBody()->getNode();
3. //log("%d %d\n", nodeA->getTag(), nodeB->getTag());
4. **if** (nodeA->getTag() == BALL) {
5. **if** (nodeB->getTag() == BOX)
6. nodeB->removeFromParentAndCleanup(**true**);
7. **else** **if** (nodeB->getTag() == SHIP)
8. GameOver();
9. }
11. **else** **if** (nodeB->getTag() == BALL) {
12. **if** (nodeA->getTag() == BOX)
13. nodeA->removeFromParentAndCleanup(**true**);
14. **else** **if** (nodeA->getTag() == SHIP)
15. GameOver();
16. }
17. 关键步骤截图





1. 亮点与改进

实现球的粒子效果：

1. meteorBall = ParticleMeteor::create();
2. meteorBall->setPosition(ball->getPosition());
3. meteorBall->setDuration(-1);

并时刻更新粒子效果的位置，在update函数中进行设置，使其与始终跟随球运动。

1. 遇到的问题

问题一：

球和板子的固定，使用PhysicsJointPin对求和板子进行固定，将其固定到一个钉子上（及某个点上）：在代码中我将两者固定到了ball->getPosition()上，也可以固定到其锚点上。

1. joint1= PhysicsJointPin::construct(
2. ball->getPhysicsBody(),player->getPhysicsBody(), ball->getPosition()
3. );

问题二：

由于板子是静态刚体，所以其运动时会出界，解决方法是在update函数中检测板块的位置，让其始终处于界面内：

1. **if** (player->getPosition().x >= visibleSize.width - player->getContentSize().width / 20 && currDir=="Right")
2. player->getPhysicsBody()->setVelocity(Vec2(0, 0));
4. **if**(player->getPosition().x <= player->getContentSize().width / 20 && currDir=="Left")
5. player->getPhysicsBody()->setVelocity(Vec2(0, 0));