Con Chim Điên Phần 2

# Quay chim

Nhìn hấp dẫn hơn thì ta sẽ cho con chim có 2 trạng thái. Hoặc là quay chổng đầu xuống đất khi rơi hoặc quay hếch mỏ lên trên khi chạm tay.

Quay được điều khiển bởi vận tốc trục Y. Trong game, khi chim rớt thì gia tốc sẽ tăng do trọng lực (có nghĩa là vận tốc tăng khi rớt xuống).

Cho 2 chỉ số vận tốc cận trên và cận dưới, khi vận tốc giảm đạt cận dưới (vận tốc giảm do ráng bay lên) thì chim quay ngược chiều kim đồng hồ (hếch mỏ lên). Khi vận tốc tăng đạt cận trên (vận tốc tăng do rớt xuống) thì chim quay theo chiều kim đồng hồ (chổng đầu xuống). Ta cũng không quay cho đến khi bắt đầu tăng tốc 1 chút.

Ngoài ra ta cũng không cho chim vỗ cánh khi bị rơi xuống. Ta sẽ viết 2 hàm có dạng

public boolean isFalling() {  
    return velocity.y > 110;  
}  
public boolean shouldntFlap() {  
    return velocity.y > 70;  
}

Một hàm khi đạt tốc độ >110 sẽ trả về true để biết là rơi xuống để quay đầu xuống, và hàm khi đạt >70 để không vẫy cánh nữa

1. Mở file Bird.java lên. Đầu tiên thêm 2 hàm mới như sau:

**public** **boolean** isFalling() {

**return** velocity.y > 110;

}

**public** **boolean** shouldntFlap() {

**return** velocity.y > 70;

}

1. Sau đó trong hàm update, bên dưới ta thêm như sau:

// quay nguoc chieu kim dong ho (hech mo len)

**if** (velocity.y < 0) {

rotation -= 600 \* delta;

**if** (rotation < -20) {

rotation = -20;

}

}

// quay nguoc chieu kim dong ho (chong dau xuong)

**if** (isFalling()) {

rotation += 480 \* delta;

**if** (rotation > 90) {

rotation = 90;

}

}

1. Quay về GameRenderer.java. Trong hàm render, tạm thời chặn phần vẽ chim kiểu cũ và vẽ lại kiểu mới như code dưới (ta con sửa nữa chứ chưa xong đâu).

// ve chim tai theo toa do. truy xuat tu AssetLoader

// truyen bien runTime de lay frame hien tai

/\*batcher.draw(AssetLoader.birdAnimation.getKeyFrame(runTime),

bird.getX(), bird.getY(), bird.getWidth(), bird.getHeight());

\*/

**if** (bird.shouldntFlap()) {

batcher.draw(AssetLoader.*birdAnimation*.getKeyFrame(1), bird.getX(), bird.getY(),

bird.getWidth() / 2.0f, bird.getHeight() / 2.0f,

bird.getWidth(), bird.getHeight(), 1, 1, bird.getRotation());

} **else** {

batcher.draw(AssetLoader.*birdAnimation*.getKeyFrame(runTime), bird.getX(),

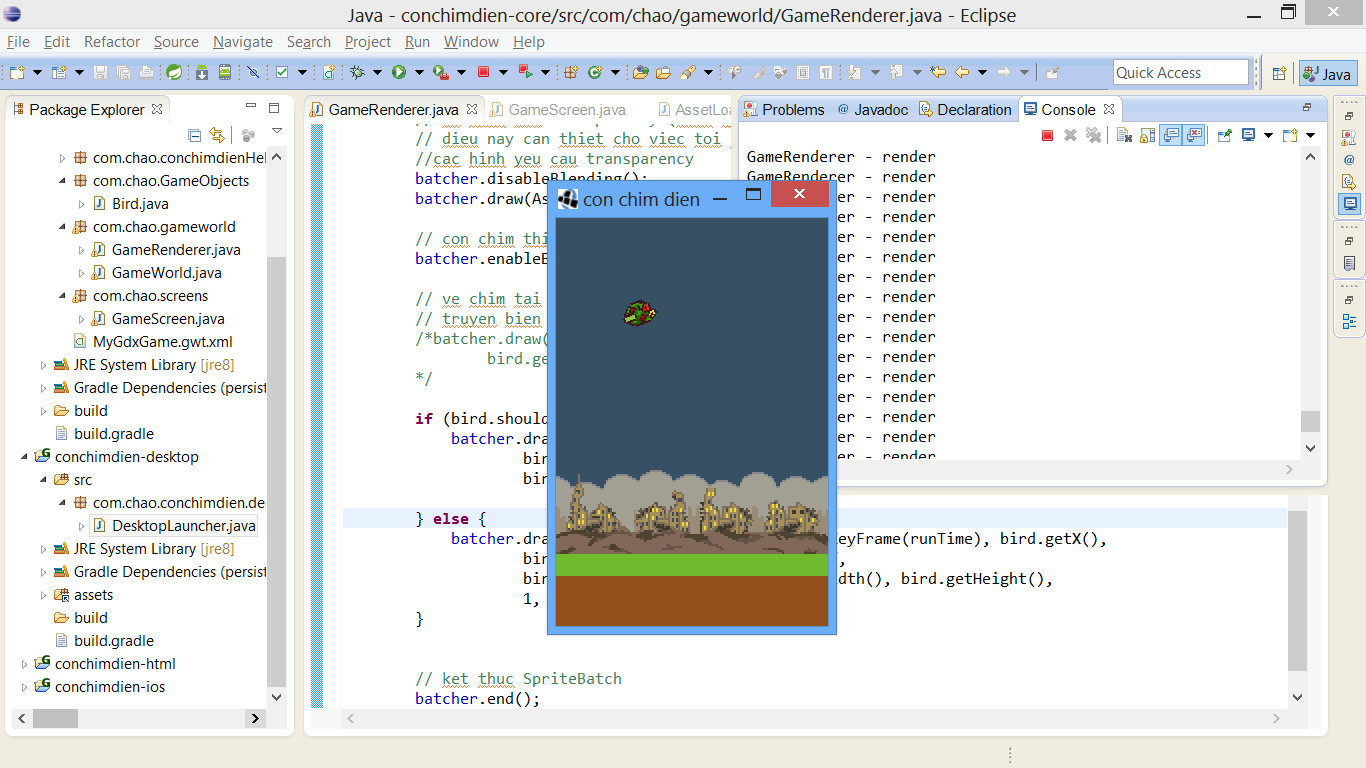
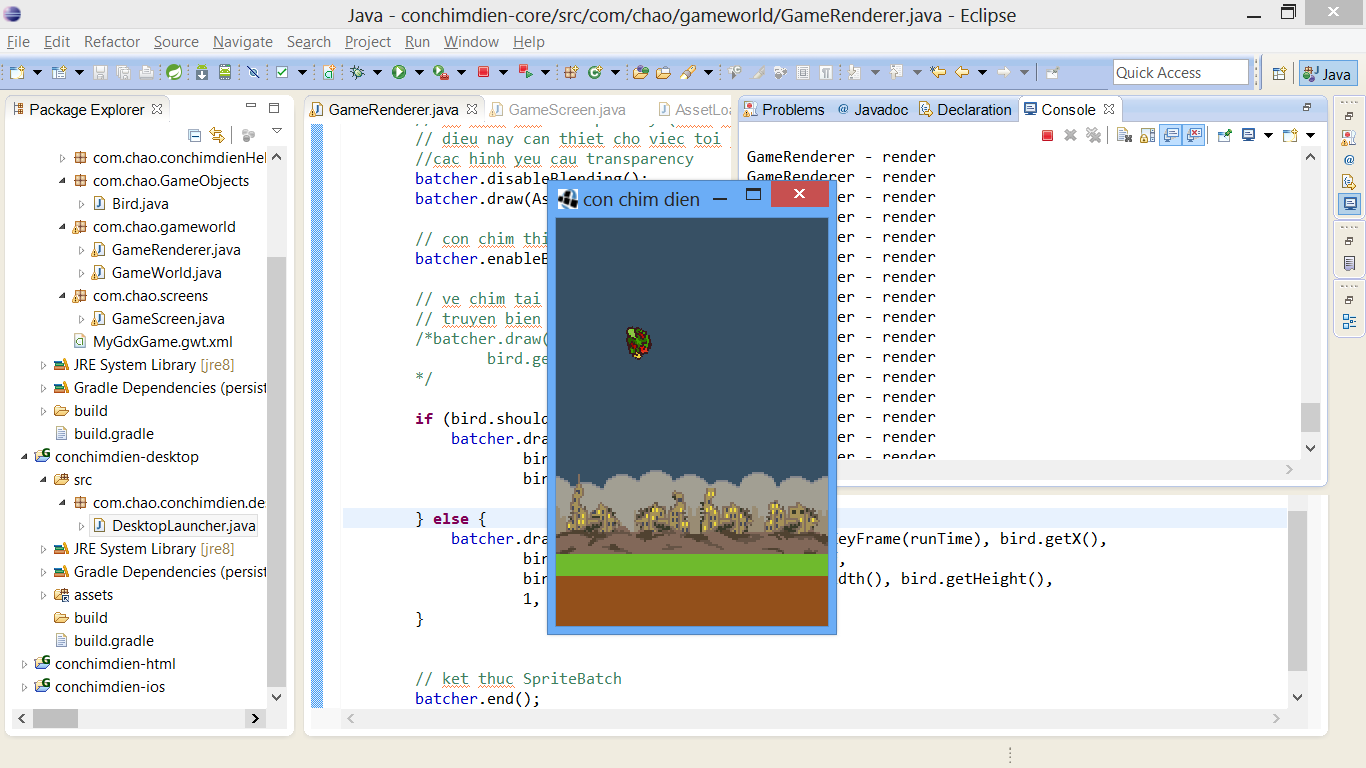
bird.getY(), bird.getWidth() / 2.0f,

bird.getHeight() / 2.0f, bird.getWidth(), bird.getHeight(),

1, 1, bird.getRotation());

}

1. Chạy file DesktopLauncher.java trong project destop và click chuột để xem chim vỗ cánh, chim chổng đầu, chim hếch mỏ.

## Tạo lớp để cuộn màn hình (Scrollable Classes)

Cỏ và đường ống sẽ di chuyển từ bên trái qua. Ta sẽ hiện khoảng 3 ống. Mỗi khi ống hoặc cỏ chạy qua (scrollable) và không còn nhìn thấy nữa ta sẽ gán nó ra sau đuôi của các đối tượng và tiếp tục cuộn tiếp. Thật ra cỏ và đường ống có cách cuộn khá giống nhau do đó ta sẽ tạo ra một lớp phục vụ cho việc cuộn là Scrollable class và cho lớp cỏ và lớp đường ống kế thừa từ nó.

Để chạy các đường ống thật sự ta chỉ cần 3 ống, ống 1, ống 2 , và ống 3. Khi ống 1 khuất khỏi màn hình nó sẽ được thiết lập lại để nằm sau ống 3 và tương tự ống 2 thiết lập lại sau ống 1 và cứ thế…

Chúng ta sẽ tạo ra một lớp Scrollable với các biến là position (vị trí), velocity (vận tốc), width, height (rộng, cao), isScrolledLeft (biến Boolean để lưu trạng thái là còn hiển thị hay không để có thể thiết lập lại). Các phương thức gồm cập nhật và thiết lập lại.

1. Trong project core, trong package GameObjects tạo một class mới tên Scrollable.java có mã như sau:

**public** **class** Scrollable {

**protected** Vector2 position;

**protected** Vector2 velocity;

**protected** **int** width;

**protected** **int** height;

**protected** **boolean** isScrolledLeft;

**public** Scrollable(**float** x, **float** y, **int** width, **int** height, **float** scrollSpeed) {

position = **new** Vector2(x, y);

velocity = **new** Vector2(scrollSpeed, 0);

**this**.width = width;

**this**.height = height;

isScrolledLeft = **false**;

}

**public** **void** update(**float** delta) {

position.add(velocity.cpy().scl(delta));

// neu doi tuong Scrollable khong con visible:

**if** (position.x + width < 0) {

isScrolledLeft = **true**;

}

}

// Reset: nen Override trong class con de co nhieu hanh vi hon

**public** **void** reset(**float** newX) {

position.x = newX;

isScrolledLeft = **false**;

}

**public** **boolean** isScrolledLeft() {

**return** isScrolledLeft;

}

**public** **float** getTailX() {

**return** position.x + width;

}

**public** **float** getX() {

**return** position.x;

}

**public** **float** getY() {

**return** position.y;

}

**public** **int** getWidth() {

**return** width;

}

**public** **int** getHeight() {

**return** height;

}

}

# Tạo đường ống

1. Tạo một class mới tên Pipe có mã như sau:

**public** **class** Pipe **extends** Scrollable {

**private** Random r;

**public** Pipe(**float** x, **float** y, **int** width, **int** height, **float** scrollSpeed) {

**super**(x, y, width, height, scrollSpeed);

// Initialize a Random object for Random number generation

r = **new** Random();

}

@Override

**public** **void** reset(**float** newX) {

//goi phuong thuc reset trong lop cha(Scrollable)

**super**.reset(newX);

// doi chieu cao bang 1 so ngau nhien

height = r.nextInt(90) + 15;

}

}

# Tạo cỏ

1. Tạo một class mới tên Grass có mã như sau:

**public** **class** Grass **extends** Scrollable {

**public** Grass(**float** x, **float** y, **int** width, **int** height, **float** scrollSpeed) {

**super**(x, y, width, height, scrollSpeed);

}

}

# Tạo phần quản lý các đối tượng

1. Tạo một class mới tên và có nội dung như sau (khi đó tạo ScrollHandler trong GameWorld sẽ tạo ra luôn 5 đối tượng, render 5 đối tượng trong GameRender cũng bằng ScrollHandler:

**public** **class** ScrollHandler {

// ScrollHandler se tao ra 5 đoi tuong cho game

**private** Grass frontGrass, backGrass; //co tren, co duoi

**private** Pipe pipe1, pipe2, pipe3; //3 duong ong

//hang so de quyet dinh khoang cach de cuon ngoai ra no con

//quyet dinh khoang cach giua 2 duong ong

**public** **static** **final** **int** *SCROLL\_SPEED* = -59;

**public** **static** **final** **int** *PIPE\_GAP* = 49;

**public** ScrollHandler(**float** yPos) {

frontGrass = **new** Grass(0, yPos, 143, 11, *SCROLL\_SPEED*);

backGrass = **new** Grass(frontGrass.getTailX(), yPos, 143, 11,

*SCROLL\_SPEED*);

pipe1 = **new** Pipe(210, 0, 22, 60, *SCROLL\_SPEED*);

pipe2 = **new** Pipe(pipe1.getTailX() + *PIPE\_GAP*, 0, 22, 70, *SCROLL\_SPEED*);

pipe3 = **new** Pipe(pipe2.getTailX() + *PIPE\_GAP*, 0, 22, 60, *SCROLL\_SPEED*);

}

**public** **void** update(**float** delta) {

// cap nhat cac doi tuong

frontGrass.update(delta);

backGrass.update(delta);

pipe1.update(delta);

pipe2.update(delta);

pipe3.update(delta);

// kiem tra 1 duong ong da roi khoi man hinh thi cap nhap no ve sau

**if** (pipe1.isScrolledLeft()) {

pipe1.reset(pipe3.getTailX() + *PIPE\_GAP*);

} **else** **if** (pipe2.isScrolledLeft()) {

pipe2.reset(pipe1.getTailX() + *PIPE\_GAP*);

} **else** **if** (pipe3.isScrolledLeft()) {

pipe3.reset(pipe2.getTailX() + *PIPE\_GAP*);

}

**if** (frontGrass.isScrolledLeft()) {

frontGrass.reset(backGrass.getTailX());

} **else** **if** (backGrass.isScrolledLeft()) {

backGrass.reset(frontGrass.getTailX());

}

}

**public** Grass getFrontGrass() {

**return** frontGrass;

}

**public** Grass getBackGrass() {

**return** backGrass;

}

**public** Pipe getPipe1() {

**return** pipe1;

}

**public** Pipe getPipe2() {

**return** pipe2;

}

**public** Pipe getPipe3() {

**return** pipe3;

}

}

1. Mở file GameWorld.java khai báo 1 biến toàn cục:

**private** ScrollHandler scroller;

1. Trong hàm tạo của GameWorld ta thêm phần tạo scroller:

// Co nen bat dau 66 pixels duoi midPointY

scroller = **new** ScrollHandler(midPointY + 66);

1. Trong hàm update gọi update scroller

scroller.update(delta);

1. Thêm một hàm mới getScroller trả về scroller

**public** ScrollHandler getScroller() {

**return** scroller;

}

# Tối ưu hóa GameRenderer và render ScrollHandler.

Quay lại GameRenderer ta thấy rằng mã lệnh viết không được tốt cho hiệu suất. Dễ thấy nhất là trong hàm render ta tạo ra con chim, (Bird bird=myWord.getBrid()). Khai báo cục bộ như thế này là nó phải lấy liên tục (60 lần / giây do đây là hàm render liên tục), do đó ta nên khai báo toàn cục và lấy nó 1 lần ở hàm tạo, ta cũng sẽ làm tương tự vậy cho các đối tượng khác.

1. Mở file GameRenderer.java lên đầu tiên khai báo thêm các biến toàn cục. (Chú ý, biến chim ta đưa thành toàn cục để tối ứu hiệu suất)

//Game Object

**private** Bird bird;

**private** ScrollHandler scroller;

**private** Grass frontGrass, backGrass;

**private** Pipe pipe1, pipe2, pipe3;

// Game Assets

**private** TextureRegion bg, grass;

**private** Animation birdAnimation;

**private** TextureRegion birdMid, birdDown, birdUp;

**private** TextureRegion skullUp, skullDown, bar;

1. Xây một hàm mới initGameObjects. Hàm giúp khởi tạo các đối tượng.

**private** **void** initGameObjects() {

bird = myWorld.getBird();

scroller = myWorld.getScroller();

frontGrass = scroller.getFrontGrass();

backGrass = scroller.getBackGrass();

pipe1 = scroller.getPipe1();

pipe2 = scroller.getPipe2();

pipe3 = scroller.getPipe3();

}

1. Một hàm mới tên initAssets

**private** **void** initAssets() {

bg = AssetLoader.*bg*;

grass = AssetLoader.*grass*;

birdAnimation = AssetLoader.*birdAnimation*;

birdMid = AssetLoader.*bird*;

birdDown = AssetLoader.*birdDown*;

birdUp = AssetLoader.*birdUp*;

skullUp = AssetLoader.*skullUp*;

skullDown = AssetLoader.*skullDown*;

bar = AssetLoader.*bar*;

}

1. Xây tiếp 3 hàm để vẽ cỏ, vẽ đầu lâu và đường ống.

**private** **void** drawGrass() {

// Draw the grass

batcher.draw(grass, frontGrass.getX(), frontGrass.getY(),

frontGrass.getWidth(), frontGrass.getHeight());

batcher.draw(grass, backGrass.getX(), backGrass.getY(),

backGrass.getWidth(), backGrass.getHeight());

}

**private** **void** drawSkulls() {

// Temporary code! Sorry about the mess :)

// We will fix this when we finish the Pipe class.

batcher.draw(skullUp, pipe1.getX() - 1,

pipe1.getY() + pipe1.getHeight() - 14, 24, 14);

batcher.draw(skullDown, pipe1.getX() - 1,

pipe1.getY() + pipe1.getHeight() + 45, 24, 14);

batcher.draw(skullUp, pipe2.getX() - 1,

pipe2.getY() + pipe2.getHeight() - 14, 24, 14);

batcher.draw(skullDown, pipe2.getX() - 1,

pipe2.getY() + pipe2.getHeight() + 45, 24, 14);

batcher.draw(skullUp, pipe3.getX() - 1,

pipe3.getY() + pipe3.getHeight() - 14, 24, 14);

batcher.draw(skullDown, pipe3.getX() - 1,

pipe3.getY() + pipe3.getHeight() + 45, 24, 14);

}

**private** **void** drawPipes() {

// Temporary code! Sorry about the mess :)

// We will fix this when we finish the Pipe class.

batcher.draw(bar, pipe1.getX(), pipe1.getY(), pipe1.getWidth(),

pipe1.getHeight());

batcher.draw(bar, pipe1.getX(), pipe1.getY() + pipe1.getHeight() + 45,

pipe1.getWidth(), midPointY + 66 - (pipe1.getHeight() + 45));

batcher.draw(bar, pipe2.getX(), pipe2.getY(), pipe2.getWidth(),

pipe2.getHeight());

batcher.draw(bar, pipe2.getX(), pipe2.getY() + pipe2.getHeight() + 45,

pipe2.getWidth(), midPointY + 66 - (pipe2.getHeight() + 45));

batcher.draw(bar, pipe3.getX(), pipe3.getY(), pipe3.getWidth(),

pipe3.getHeight());

batcher.draw(bar, pipe3.getX(), pipe3.getY() + pipe3.getHeight() + 45,

pipe3.getWidth(), midPointY + 66 - (pipe3.getHeight() + 45));

}

1. Trong hàm tạo của GameRenderer thêm 2 lời gọi hàm để khởi tạo các đối tượng

initGameObjects();

initAssets();

1. Trong hàm render. Bỏ khởi tạo con chim đi, sửa lại để vẽ thêm như sau (các chổ thay đổi đánh số 1,2,3)

**public** **void** render(**float** runTime) {

System.*out*.println("GameRenderer - render");

Gdx.*gl*.glClearColor(0, 0, 0, 1);

Gdx.*gl*.glClear(GL20.*GL\_COLOR\_BUFFER\_BIT*);

// bat dau ShapeRenderer

shapeRenderer.begin(ShapeType.*Filled*);

// ve Background color

shapeRenderer.setColor(55 / 255.0f, 80 / 255.0f, 100 / 255.0f, 1);

shapeRenderer.rect(0, 0, 136, midPointY + 66);

// ve Grass (co)

shapeRenderer.setColor(111 / 255.0f, 186 / 255.0f, 45 / 255.0f, 1);

shapeRenderer.rect(0, midPointY + 66, 136, 11);

// ve Dirt (ve dam nha cua va may)

shapeRenderer.setColor(147 / 255.0f, 80 / 255.0f, 27 / 255.0f, 1);

shapeRenderer.rect(0, midPointY + 77, 136, 52);

// ket thuc ShapeRenderer

shapeRenderer.end();

// bat dau SpriteBatch

batcher.begin();

// vo hieu hoa transparency (tinh trong suot)

// dieu nay can thiet cho viec toi uu toc do khi khong can phai ve

//cac hinh yeu cau transparency

batcher.disableBlending();

batcher.draw(bg, 0, midPointY + 23, 136, 43);

// con chim thi can transparency

batcher.enableBlending();

// 1. ve Grass

drawGrass();

// 2. ve Pipes

drawPipes();

// 3. ve Skulls

drawSkulls();

**if** (bird.shouldntFlap()) {

batcher.draw(birdMid, bird.getX(), bird.getY(),

bird.getWidth() / 2.0f, bird.getHeight() / 2.0f,

bird.getWidth(), bird.getHeight(), 1, 1, bird.getRotation());

} **else** {

batcher.draw(birdAnimation.getKeyFrame(runTime), bird.getX(),

bird.getY(), bird.getWidth() / 2.0f,

bird.getHeight() / 2.0f, bird.getWidth(), bird.getHeight(),

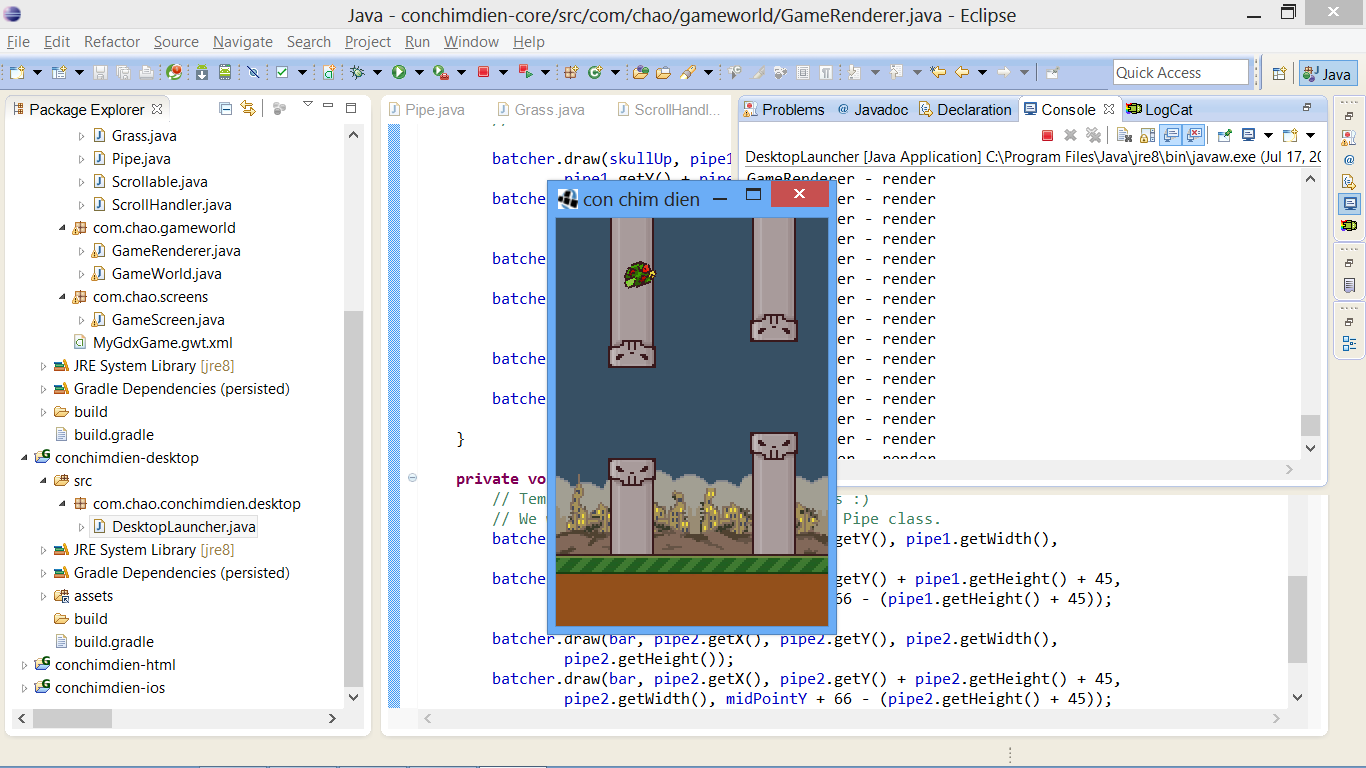
1, 1, bird.getRotation());

}

batcher.end();

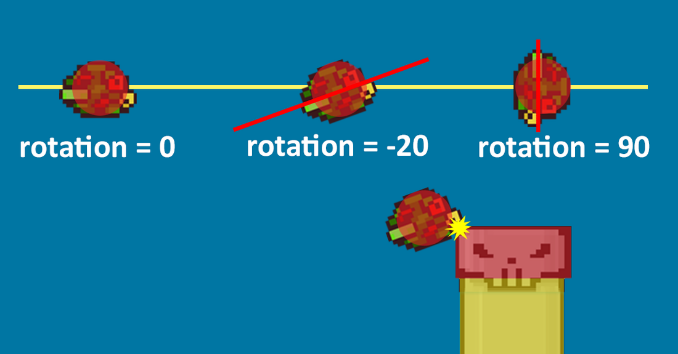
}

1. Chạy thử và thấy kết quả (thấy rằng chim bay đường chim, ống chạy đường ống).



# Xem xét va chạm

Tiếp theo ta sẽ xét va chạm. Có thể cho va chạm giữa hình chữ nhật chim với hình chữ nhật các đối tượng khác, nhưng chim quay lên quay xuống do đó ta chọn giải pháp con chim là hình tròn để thuận lợi hơn, vậy sẽ là hình tròn va chạm với hình chữ nhật.



# Tạo hình tròn ranh giới trong chim để xét va chạm

1. Mở lớp Bird.java lên, khai báo 1 biến Circle toàn cục như sau:

**private** Circle boundingCircle;

1. Trong hàm tạo bird khởi tạo đối tượng Circle

boundingCircle = **new** Circle();

1. Trong hàm update gán tâm và bán kính của hình tròn phủ lên chim sau lệnh position.add

position.add(velocity.cpy().scl(delta));

// gan tam hinh trong la (9, 6)

// gan ban kinh la 6.5f;

boundingCircle.set(position.x + 9, position.y + 6, 6.5f);

1. Thêm 1 function

**public** Circle getBoundingCircle() {

**return** boundingCircle;

}

1. Để test xem hình tròn vẽ đúng chưa ta mở file GameRenderer.java và thêm đoạn mã bên dưới vào dưới cùng của hàm render

shapeRenderer.begin(ShapeType.*Filled*);

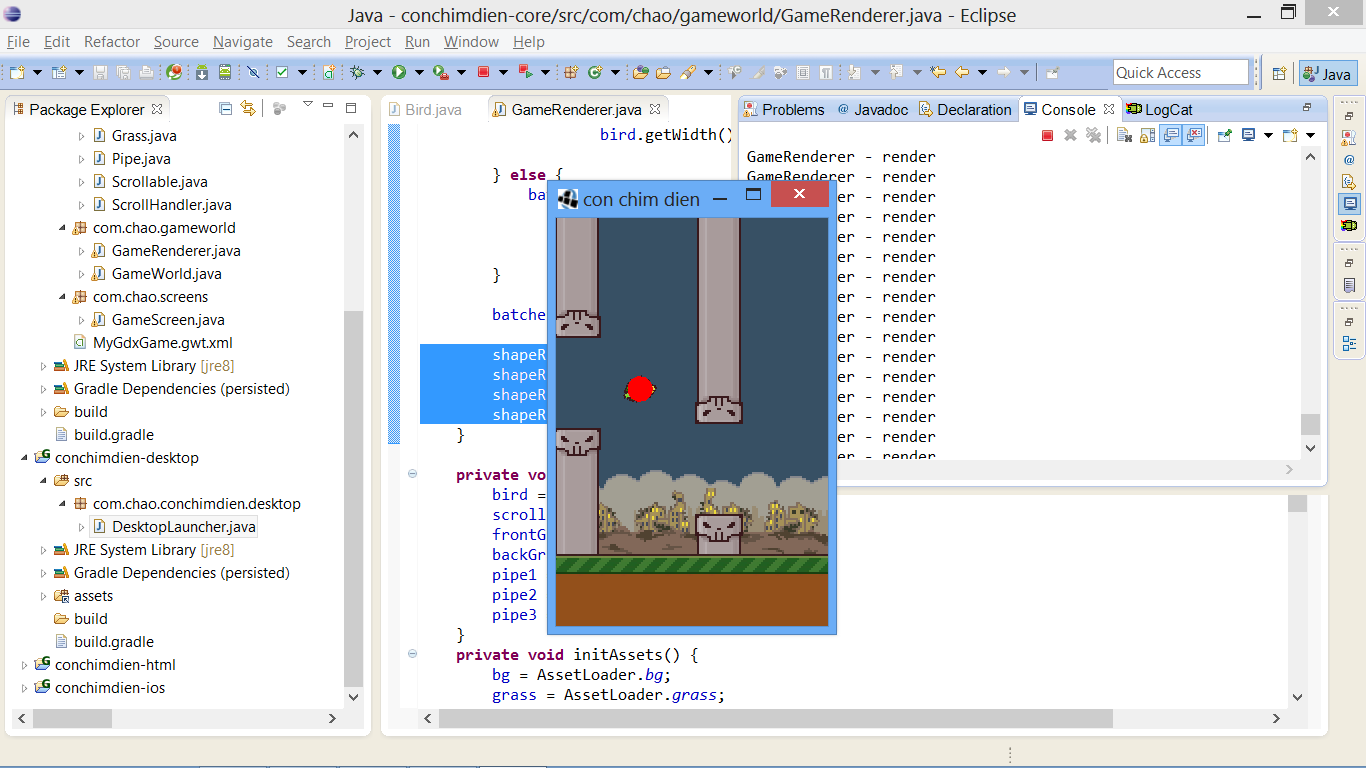
shapeRenderer.setColor(Color.*RED*);

shapeRenderer.circle(bird.getBoundingCircle().x,

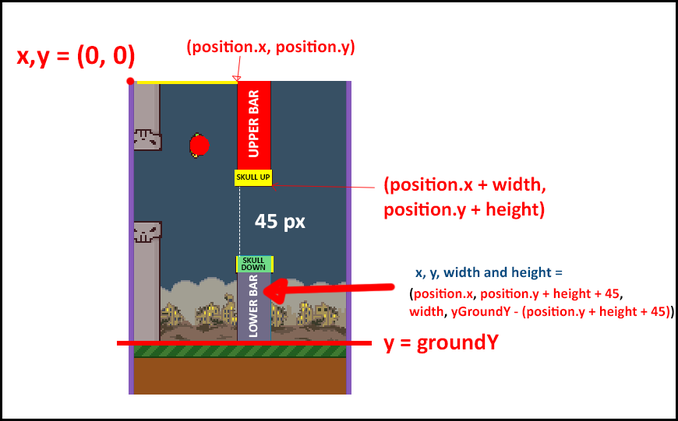
bird.getBoundingCircle().y, bird.getBoundingCircle().radius);

shapeRenderer.end();

1. Chạy file DesktopLauncher.java để thấy có 1 hình tròn đỏ đè lên đại ca chim



# Tạo hình chữ nhật ranh giới trên các đối tượng đường ống



1. Mở lớp Pipe.java khai báo thêm các biến toàn cục và các hằng số

**private** Rectangle skullUp, skullDown, barUp, barDown;

**public** **static** **final** **int** *VERTICAL\_GAP* = 45;

**public** **static** **final** **int** *SKULL\_WIDTH* = 24;

**public** **static** **final** **int** *SKULL\_HEIGHT* = 11;

**private** **float** groundY;

1. Trong hàm tạo nhận thêm biến groudY và khởi tạo các đối tượng khác như sau:

**public** Pipe(**float** x, **float** y, **int** width, **int** height, **float** scrollSpeed,**float** groundY) {

**super**(x, y, width, height, scrollSpeed);

r = **new** Random();

skullUp = **new** Rectangle();

skullDown = **new** Rectangle();

barUp = **new** Rectangle();

barDown = **new** Rectangle();

**this**.groundY = groundY;

}

1. Override lên hàm update để vẽ hình chữ nhật.

@Override

**public** **void** update(**float** delta) {

// goi update trong superclass (Scrollable)

**super**.update(delta);

// set() giúp thiet lap goc tren trai va rong, cao cua hinh chu nhat

barUp.set(position.x, position.y, width, height);

barDown.set(position.x, position.y + height + *VERTICAL\_GAP*, width,

groundY - (position.y + height + *VERTICAL\_GAP*));

// chieu rong skull la 24. bar chi 22 do do skull can dich sang trai 1 pixel

skullUp.set(position.x - (*SKULL\_WIDTH* - width) / 2, position.y + height

- *SKULL\_HEIGHT*, *SKULL\_WIDTH*, *SKULL\_HEIGHT*);

skullDown.set(position.x - (*SKULL\_WIDTH* - width) / 2, barDown.y,

*SKULL\_WIDTH*, *SKULL\_HEIGHT*);

}

1. Thêm 4 hàm nữa

**public** Rectangle getSkullUp() {

**return** skullUp;

}

**public** Rectangle getSkullDown() {

**return** skullDown;

}

**public** Rectangle getBarUp() {

**return** barUp;

}

**public** Rectangle getBarDown() {

**return** barDown;

}

# Cập nhật ScrollHandler

1. Do ta sửa ở trên nên các pipe của ScrollHandler sẽ bị lỗi. Mở ScrollHandler trong hàm tạo truyền thêm vào biến yPos cho các pipe

pipe1 = **new** Pipe(210, 0, 22, 60, *SCROLL\_SPEED*,yPos);

pipe2 = **new** Pipe(pipe1.getTailX() + *PIPE\_GAP*, 0, 22, 70, *SCROLL\_SPEED*,yPos);

pipe3 = **new** Pipe(pipe2.getTailX() + *PIPE\_GAP*, 0, 22, 60, *SCROLL\_SPEED*,yPos);

Cập nhật GameRenderer

1. Mở GameRenderer, trong hàm render, kéo xuống dưới và thêm đoạn mã vào trước lệnh shapeRenderer.end();(ở cuối).

// Bar up for pipes 1 2 and 3

shapeRenderer.rect(pipe1.getBarUp().x, pipe1.getBarUp().y,

pipe1.getBarUp().width, pipe1.getBarUp().height);

shapeRenderer.rect(pipe2.getBarUp().x, pipe2.getBarUp().y,

pipe2.getBarUp().width, pipe2.getBarUp().height);

shapeRenderer.rect(pipe3.getBarUp().x, pipe3.getBarUp().y,

pipe3.getBarUp().width, pipe3.getBarUp().height);

// Bar down for pipes 1 2 and 3

shapeRenderer.rect(pipe1.getBarDown().x, pipe1.getBarDown().y,

pipe1.getBarDown().width, pipe1.getBarDown().height);

shapeRenderer.rect(pipe2.getBarDown().x, pipe2.getBarDown().y,

pipe2.getBarDown().width, pipe2.getBarDown().height);

shapeRenderer.rect(pipe3.getBarDown().x, pipe3.getBarDown().y,

pipe3.getBarDown().width, pipe3.getBarDown().height);

// Skull up for Pipes 1 2 and 3

shapeRenderer.rect(pipe1.getSkullUp().x, pipe1.getSkullUp().y,

pipe1.getSkullUp().width, pipe1.getSkullUp().height);

shapeRenderer.rect(pipe2.getSkullUp().x, pipe2.getSkullUp().y,

pipe2.getSkullUp().width, pipe2.getSkullUp().height);

shapeRenderer.rect(pipe3.getSkullUp().x, pipe3.getSkullUp().y,

pipe3.getSkullUp().width, pipe3.getSkullUp().height);

// Skull down for Pipes 1 2 and 3

shapeRenderer.rect(pipe1.getSkullDown().x, pipe1.getSkullDown().y,

pipe1.getSkullDown().width, pipe1.getSkullDown().height);

shapeRenderer.rect(pipe2.getSkullDown().x, pipe2.getSkullDown().y,

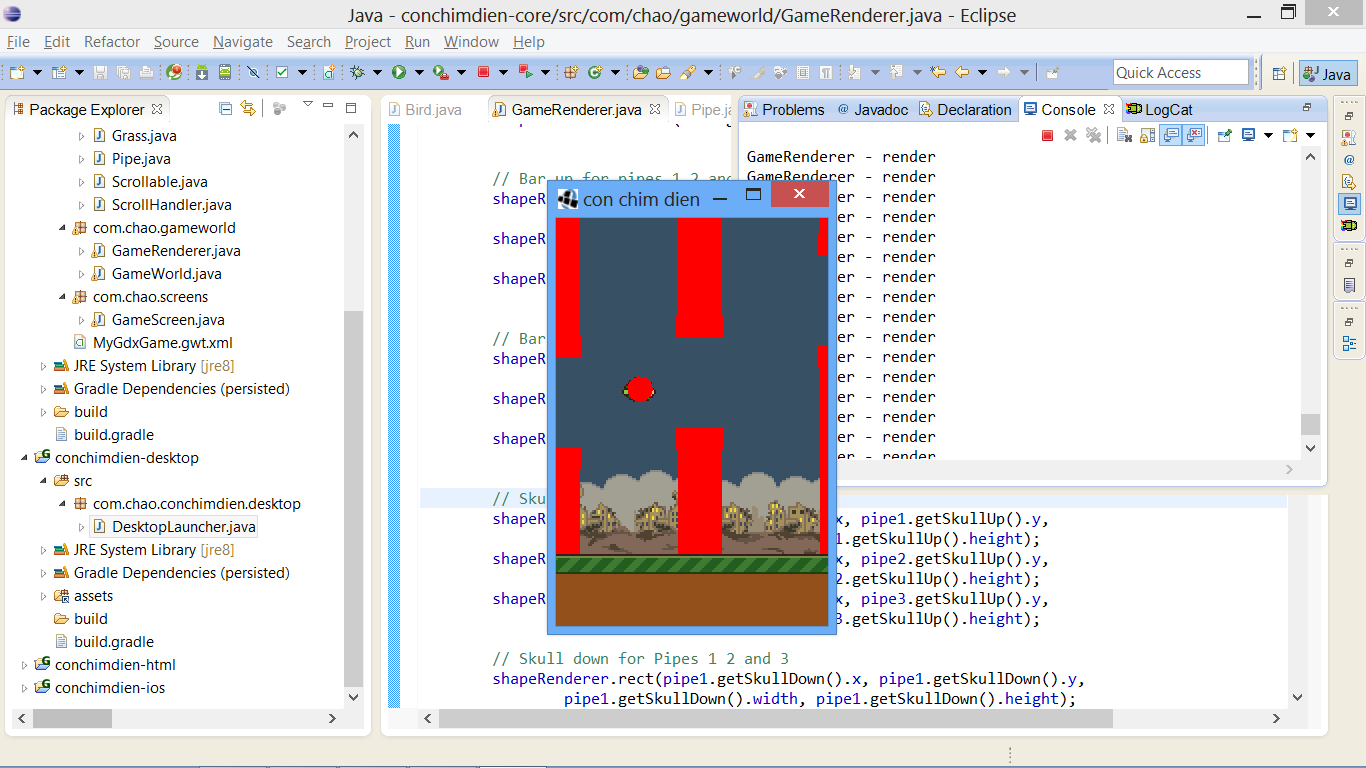
pipe2.getSkullDown().width, pipe2.getSkullDown().height);

shapeRenderer.rect(pipe3.getSkullDown().x, pipe3.getSkullDown().y,

pipe3.getSkullDown().width, pipe3.getSkullDown().height);

shapeRenderer.end(); /\* \* \*/

1. Chạy chương trình để thấy chim và ống đều đỏ như hình



# Kiểm tra va chạm

Kiểm tra va chạm đòi hỏi nhiều lớp làm việc với nhau.

ScrollHandler có thể truy cập tất cả các hình chữ nhật ranh giới do đó ta sẽ kiểm va chạm trong nó.

GameWorld cần biết va chạm để điều khiển các hành động như lấy điểm số, dừng chim, chơi âm thanh…

GameRender tương tác khi chim chết hiển thị điểm số, hiển thị màn hình chào….

1. Mở Pipe.java thêm hàm sau để xét va chạm

**public** **boolean** collides(Bird bird) {

**if** (position.x < bird.getX() + bird.getWidth()) {

**return** (Intersector.*overlaps*(bird.getBoundingCircle(), barUp)

|| Intersector.*overlaps*(bird.getBoundingCircle(), barDown)

|| Intersector.*overlaps*(bird.getBoundingCircle(), skullUp)

|| Intersector.*overlaps*(bird.getBoundingCircle(), skullDown));

}

**return** **false**;

}

1. Mở Scrollable.java lên và thêm hàm stop

**public** **void** stop() {

velocity.x = 0;

}

1. Mở ScrollHandler.java lên và thêm 2 hàm để dừng chạy và gọi va chạm

**public** **void** stop() {

frontGrass.stop();

backGrass.stop();

pipe1.stop();

pipe2.stop();

pipe3.stop();

}

**public** **boolean** collides(Bird bird) {

**return** (pipe1.collides(bird) || pipe2.collides(bird) || pipe3.collides(bird));

}

1. Mở GameWorld.java lên, trong hàm update thêm phần if để xét khi va chạm

**public** **void** update(**float** delta)

{

bird.update(delta);

scroller.update(delta);

**if** (scroller.collides(bird)) {

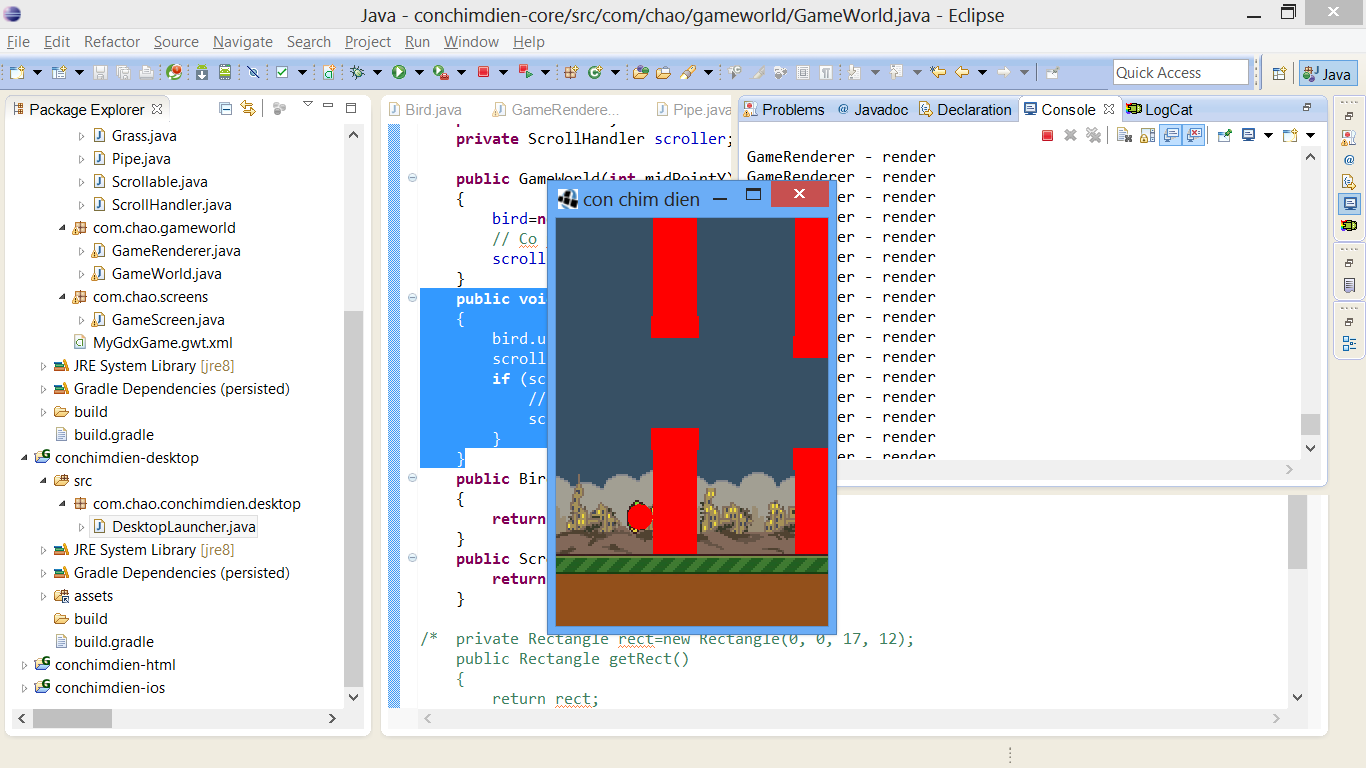
// len dia, da game over

scroller.stop();

}

}

1. Chạy để thấy khi chim đụng cột chim chổng đầu



# Dọn dẹp chiến trường

1. Sau khi thấy đã chạy ổn ta bỏ các hình vẽ đỏ đỏ của chim và cột. Mở File GameRender, trong hàm update bỏ các phần lệnh vẽ hình tròn và hình chữ nhật ở cuối

**public** **void** render(**float** runTime) {

………………………………………….

**else** {

batcher.draw(birdAnimation.getKeyFrame(runTime), bird.getX(),

bird.getY(), bird.getWidth() / 2.0f,

bird.getHeight() / 2.0f, bird.getWidth(), bird.getHeight(),

1, 1, bird.getRotation());

}

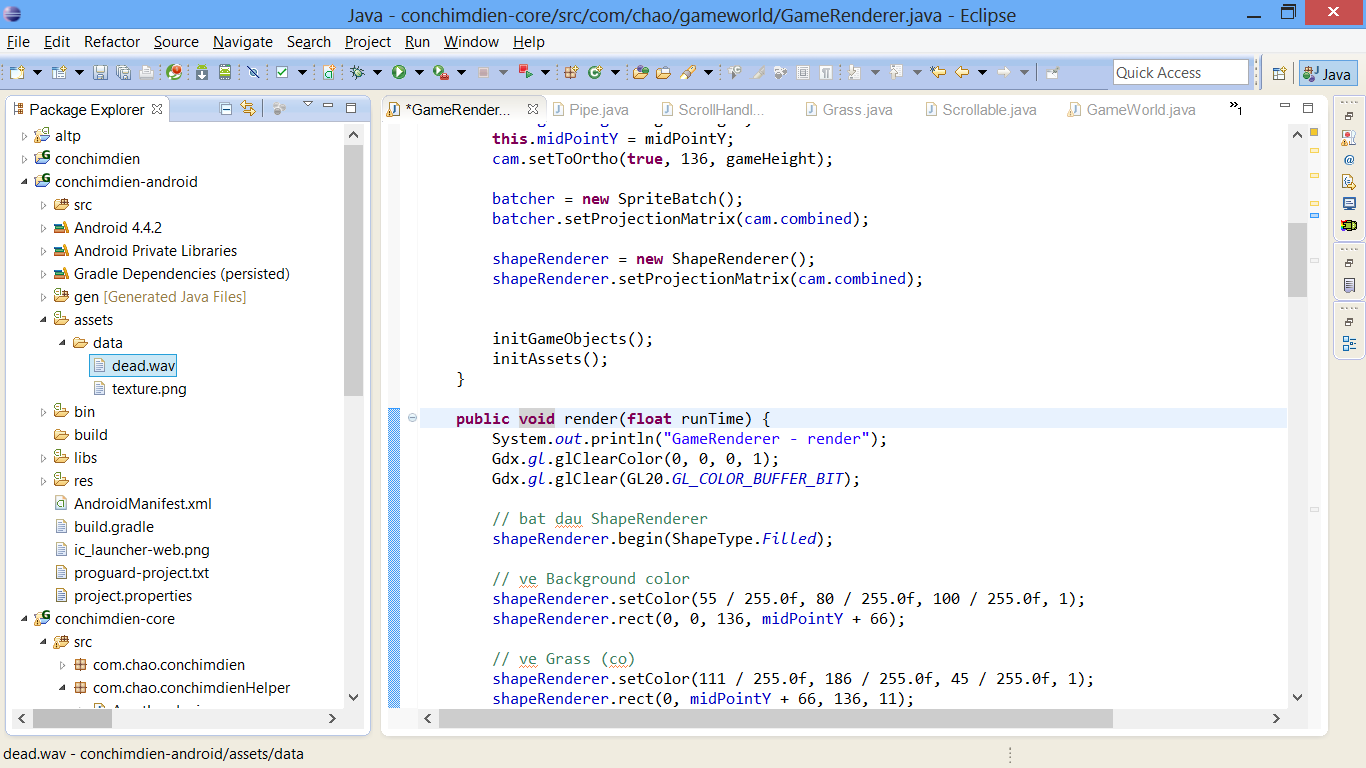
batcher.end();

/\* bo phan nay \*/

}

# Thêm âm thanh

1. Chép file âm thanh dead.wav vào thư mục assets/data trong project android.



1. Mở file AssetLoader và khai báo biến toàn cục

**public** **static** Sound *dead*;

1. Trong hàm load thêm lệnh

*dead* = Gdx.*audio*.newSound(Gdx.*files*.internal("data/dead.wav"));

1. Mở GameWorld.java lên khai báo biến toàn cục

**private** **boolean** isAlive = **true**;

1. Cập nhật phần if trong hàm update như sau

**if** (scroller.collides(bird) && isAlive) {

scroller.stop();

AssetLoader.*dead*.play();

isAlive = **false**;

}

1. Chạy và khi chim chết nghe cái “chéo”

Đến đây là hoàn tất ngày thứ 8 , các bạn xem tiếp ở địa chỉ : http://www.kilobolt.com/