

Trường Đại học Giao Thông Vận Tải

Đề tài: Lập trình Game 2D Flappy Bird

Sinh viên thực hiện: Nguyễn Nhật Minh

Mã sinh viên: 211210129

Lớp: CNTT 6 - K62

Báo cáo Bài tập lớn

Môn học: Công nghệ Java

# Giới thiệu đề tài

Flappy Bird là một trò chơi được phát hành vào năm 2013 bởi nhà phát triển Việt Nam Nguyễn Hà Đông.

Mục tiêu của trò chơi là điều hướng một con chim qua một loạt các đường ống khói bằng cách chạm vào màn hình để làm cho con chim vỗ cánh bay lên. Khi đi qua mỗi đường ống khói thành công đều được tính điểm và trò chơi được đánh giá là rất khó khi đòi hỏi người chơi phải thật kiên trì và tập trung cao độ.

# Xác định các Package trong bài

* Package SourcePackages
* Package flappybirds

Bao gồm các Class của các đối tượng cụ thể trong trò chơi

* Package pkg2dgamesframework

Bao gồm các Class của một trò chơi 2D cơ bản

* File Assets

Dùng để chứa cách hình ảnh của đối tượng đính kèm trong trò chơi

# Chức năng của các Class

* Class FlappyBirds

Đây là Class sử dụng thư viện 2D Games Framework dùng để xử lý đồ họa và hoạt ảnh của trò chơi. Trò chơi bao gồm một con chim mà người chơi điều khiển bằng phím SPACE để bay qua một loạt chướng ngại vật (ống khói) đồng thời tránh va chạm với chúng hoặc va chạm với mặt đất. Trò chơi theo dõi điểm số của người chơi dựa trên số chướng ngại vật mà chú chim đã vượt qua thành công. Đoạn code trong class này xử lý logic, hoạt ảnh và đồ họa của trò chơi, bao gồm cả việc đặt lại trò chơi và xử lý các sự kiện quan trọng

* Class Bird

Class Bird mở rộng từ class Object và có các thuộc tính sau:

- *vt (float)*: thể hiện vận tốc thẳng đứng của chim

- *isFlying (boolean)*: cho biết con chim hiện đang bay hay không

- *rect (Hình chữ nhật)*:đại diện cho hitbox hình chữ nhật của con chim

- *isLive(boolean)*: cho biết con chim còn sống hay không

Hàm tạo *Bird* có bốn đối số (x, y, w, h) để đặt vị trí và kích thước của con chim.

Lớp Bird có các phương thức sau:

- *setLive(boolean b)*: đặt giá trị của *isLive* thành giá trị boolean đã cho

- *getLive()*: trả về giá trị của *isLive*

- *getRect()*: trả về đối tượng Rectangle cho con chim

- *setVt(float vt)*: đặt giá trị của *vt* thành giá trị float đã cho

- *update(long deltaTime)*: cập nhật vị trí của con chim dựa trên vận tốc thẳng đứng và trọng lực của nó, cũng như kiểm tra xem con chim có đang bay hay không

- *fly()*: đặt giá trị của *vt* thành giá trị âm để mô phỏng chú chim đang bay lên

- *getIsFlying()*: trả về giá trị của *isFlying* để cho biết con chim hiện đang bay hay không

* Class Chimney

Đây là một Class mở rộng từ Class Objects. Nó đại diện cho một đối tượng ống khói trong trò chơi Flappy Bird

- Hàm tạo nhận các tham số x, y, chiều rộng và chiều cao và tạo một đối tượng hình chữ nhật dựa trên các tham số đó

- Phương thức *update* dùng để cập nhật vị trí của ống khói bằng cách giảm vị trí nằm ngang của nó xuống 2 đơn vị

- Phương thức *getRect* trả về đối tượng hình chữ nhật được liên kết với ống khói.

- Các phương thức *setIsBehindBird* và *getIsBehindBird* được sử dụng để xác định xem ống khói có ở phía sau con chim hay không

* Class ChimneyGroup

Đây là Class dùng để xác định một nhóm ống khói trong trò chơi. Ống khói là chướng ngại vật mà chim cần phải tránh

Lớp có các phương thức sau:

- Phương thức *getRandomY()* trả về tọa độ Y ngẫu nhiên cho ống khói nằm trong phạm vi 0-350

- Hàm tạo *ChimneyGroup()* khởi tạo một hàng gồm 6 ống khói với các ống trên và dưới xen kẽ ở các vị trí ngẫu nhiên

- Phương thức *resetChimneys()* đặt lại hàng ống khói về vị trí ban đầu của chúng

- Phương thức *update()* cập nhật vị trí của từng ống khói trong hàng đợi

- Phương thức *paint(Graphics2D g2)* vẽ các ống khói trên màn hình

Nhìn chung, lớp này có chức năng quản lý vị trí, thiết lập và hiển thị các ống khói trong trò chơi

* Class Group

Mục đích của lớp này là tạo và cập nhật hình ảnh mặt đất và vị trí trên màn hình trò chơi

Phương thức chính trong lớp này là phương thức *Update* chịu trách nhiệm di chuyển mặt đất để tạo ra ảo ảnh chuyển động trong trò chơi

Phương thức *Paint* được sử dụng để hiển thị hình ảnh mặt đất trên màn hình trò chơi

Phương thức *getYGround* được sử dụng để lấy vị trí hiện tại của mặt đất trên trục y nhằm mục đích phát hiện va chạm

Class Ground là một thành phần quan trọng của trò chơi Flappy Bird vì nó cung cấp một hệ quy chiếu để người chơi điều hướng trò chơi

* Class AFrameOnImage

Class này đại diện cho một khung hình trên một hình ảnh. Nó cho phép bạn xác định vị trí và kích thước của khung trong ảnh và cung cấp một phương pháp để vẽ khung lên đối tượng *Graphics2D*. Hình ảnh phải được cung cấp dưới dạng *BufferedImage*. Phương thức *Paint* áp dụng xoay cho khung nếu góc xoay được chỉ định. Phương pháp *VisibleRectDebug* có thể được sử dụng để vẽ một hình chữ nhật trên khung nhằm mục đích gỡ lỗi

* Class Animation

Đây là cũng là một lớp từ bộ khung trò chơi 2D. Nó được sử dụng để tạo hoạt ảnh cho các đối tượng trong trò chơi. Dưới đây là một số chi tiết:

- Lớp có một trường private gọi là *beginTime* và một trường public gọi là *mesure*. Chúng được sử dụng để theo dõi thời gian cho hoạt ảnh

- Lớp có một mảng các đối tượng *AFrameOnImage* được gọi là *frames*. Mỗi đối tượng *AFrameOnImage* đại diện cho một khung hình trong hoạt ảnh

- Lớp có các phương thức để thêm khung vào hoạt ảnh, cập nhật hoạt ảnh dựa trên thời gian và vẽ khung hiện tại của hoạt ảnh trên màn hình

- Phương thức *Update\_Me* được sử dụng để cập nhật khung hình hiện tại của hoạt ảnh dựa trên khoảng thời gian đã trôi qua kể từ lần cập nhật cuối cùng

- Phương thức *AddFrame* được sử dụng để thêm khung vào hoạt ảnh

- Phương thức *PaintAnims* được sử dụng để tô khung hình động hiện tại trên màn hình. Nó nhận các tham số cho vị trí của đối tượng, hình ảnh của đối tượng, ngữ cảnh đồ họa để vẽ lên, điểm neo để xác định vị trí của đối tượng và góc xoay cho đối tượng

* Class GameScreen

Đây là Class mở rộng từ Class JFrame và thực thi giao diện KeyListener. Nó là một lớp trừu tượng có nghĩa là nó không thể được khởi tạo nhưng có thể được mở rộng bởi một lớp khác. Lớp này được thiết kế để sử dụng làm cơ sở để tạo trò chơi 2D. Nó có một số phương thức và biến có thể được sử dụng để xử lý các chức năng liên quan đến trò chơi.

Một số phương thức quan trọng của lớp này là:

- *RegisterImage(int id, BufferedImage image)*: Phương thức này có thể được sử dụng để đăng ký một hình ảnh để sử dụng sau này trong trò chơi

- *getImageWithID(int id)*: Phương thức này truy xuất một hình ảnh có ID đã cho

- *BeginGame()*: Phương thức này khởi động GameThread chịu trách nhiệm cập nhật và hiển thị trò chơi

- *GAME\_UPDATE(long deltaTime)*: Đây là một phương thức trừu tượng phải được triển khai bởi lớp mở rộng GameScreen. Nó được gọi trong mọi vòng lặp của GameThread và có thể được sử dụng để cập nhật trạng thái trò chơi

- *GAME\_PAINT(Graphics2D g2)*: Đây là một phương thức trừu tượng cũng phải được triển khai bởi lớp mở rộng GameScreen. Nó chịu trách nhiệm hiển thị tất cả đồ họa của trò chơi

- *KEY\_ACTION(KeyEvent e, int Event)*: Phương thức trừu tượng này được sử dụng để xử lý các sự kiện chính như nhấn và nhả phím

Ngoài ra, có một số biến như *KEY\_PRESSED, KEY\_RELEASED, CUSTOM\_WIDTH, CUSTOM\_HEIGHT, MASTER\_WIDTH* và *MASTER\_HEIGHT* chứa một số dữ liệu liên quan đến trò chơi

Lớp này cung cấp một cấu trúc cơ bản để tạo các màn chơi có thể mở rộng và tùy chỉnh dựa trên các yêu cầu của một trò chơi cụ thể

* Class GameThread

Bao gồm hàm tạo lấy ngữ cảnh GameScreen và khởi tạo một số biến và luồng

- Phương thức *StartThread()* bắt đầu luồng

- Phương thức *paint()* vẽ đồ họa trên bảng điều khiển. Đầu tiên, nó lấp đầy nền bảng điều khiển bằng màu trắng, sau đó chia tỷ lệ và vẽ hình ảnh bộ đệm trên đó

- Phương thức *UpdateSize()* cập nhật kích thước của bảng điều khiển theo chiều rộng và chiều cao tùy chỉnh được xác định trong GameScreen. Nó cũng tính toán các giá trị tỷ lệ cho kích thước hình ảnh bộ đệm

- Phương thức *run()* là vòng lặp chạy ở các khung hình được chỉ định trên giây (FPS). Nó cập nhật kích thước, gọi các phương thức *GAME\_UPDATE()* và *GAME\_PAINT()* của bối cảnh GameScreen, vẽ hình ảnh bộ đệm và gọi *repaint().* Nó cũng tính toán thời gian ngủ để duy trì FPS không đổi

* Class Objects

Lớp này có các biến thể hiện riêng cho vị trí x và y của đối tượng (*posX* và *posY*), cũng như chiều rộng và chiều cao của đối tượng (*w* và *h*)

Có hai hàm tạo, một hàm đặt tất cả các biến thể hiện thành 0 và một hàm nhận các giá trị x, y, chiều rộng và chiều cao làm tham số

Lớp này cũng có các phương thức để phát hiện xung đột với các đối tượng khác bằng cách sử dụng vị trí và kích thước của chúng. Có hai kiểu của phương thức *isCollisionHappenWith()* - một kiểu chỉ lấy vị trí x và y của đối tượng kia, và một kiểu khác lấy luôn cả chiều rộng và chiều cao của nó

Có các phương thức getter và setter cho các biến thể hiện, cũng như các phương thức để tăng giá trị vị trí x và y

* Class QueueList

Đây là một sự thực thi của cấu trúc dữ liệu Hàng đợi bằng cách sử dụng danh sách được liên kết. Đây là một cách triển khai chung, nghĩa là nó có thể lưu trữ bất kỳ loại đối tượng nào trong hàng đợi

Lớp QueueList có một lớp bên trong được gọi là Element đại diện cho một nút trong danh sách được liên kết. Mỗi Element có một giá trị và tham chiếu đến phần tử tiếp theo trong danh sách

Lớp QueueList có các phương thức sau:

- *push(T t)*: Phương thức này thêm một phần tử mới vào cuối hàng đợi

- *pop()*: Phương thức này loại bỏ và trả về phần tử ở đầu hàng đợi

- *get(int id)*: Phương thức này trả về phần tử tại chỉ mục đã chỉ định trong hàng đợi

- *getSize()*: Phương thức này trả về kích thước hiện tại của hàng đợi

Việc triển khai sử dụng hai con trỏ *head* và *foot* để theo dõi các phần tử đầu tiên và cuối cùng trong hàng đợi. Khi một phần tử được đẩy vào hàng đợi, nó sẽ trở thành *foot* mới. Khi một phần tử được bật ra khỏi hàng đợi, con trỏ *đầu* được cập nhật thành phần tử tiếp theo

Việc triển khai Hàng đợi bằng cách sử dụng danh sách được liên kết này rất hiệu quả để thêm và xóa các phần tử khỏi hàng đợi và cho phép thay đổi kích thước động