HỌC VIỆN HÀNG KHÔNG VIỆT NAM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CƠ SỞ NGÀNH

HỌC KỲ 3 – NĂM HỌC: 2022 - 2023

MÃ LỚP HỌC PHẦN: 010100085102

GVHD: Huỳnh Thanh Sơn, Tô Bá Lâm

Nhóm sinh viên thực hiện: Nhóm 03

Họ tên	*Mã sinh viên*
1. Lê Cao Tấn Lộc	2154810086
2. Nguyễn Thị Thúy Hà	2154810061
3. Trịnh Vinh Qui	2154810110
4. Phan Tường Bảo Trâm	2154810077
5. Đặng Thị Hải Hà	2154810052

TPHCM, tháng 07 năm 2023

NHẬN XÉT CỦA GIẢNG VIÊN:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

LỜI CẢM ƠN

Xin chân thành cảm ơn Học viện hàng không Việt Nam và khoa Công nghệ thông tin đã đưa môn Đồ án cơ sở ngành cũng như tạo điều kiện và hỗ trợ suốt quá trình học tập của chúng em. Đồng thời nhóm em cũng không quên gửi lời cảm ơn đến thầy Huỳnh Thanh Sơn cùng với thầy Tô Bá Lâm đã đồng hành và tận tình dẫn dắt, truyền dạy những kiến thức bổ ích, những kinh nghiệm quý báu cho chúng em trong quá trình nghiên cứu đề tài này; đây chắc chắn là những kiến thức quan trọng, là hành trang để nhóm em có thể vững bước sau này. Nhờ những kiến thức bổ ích đó cùng với sự dẫn dắt và chỉ bảo tận tình của các thầy đã giúp đồ án của nhóm hoàn thành thuận lợi. Đồng thời, cảm ơn những thành viên trong nhóm đã nhiệt tình trao đổi, đóng góp ý kiến giúp cho bài báo cáo hoàn thành đúng thời gian quy định. Tuy đã có nhiều cố gắn nhưng do kiến thức khá sâu rộng và thời gian nghiên cứu có hạn nên đồ án của nhóm em sẽ không tránh khỏi những sai sót. Chúng em rất mong nhận được sự góp ý của các thầy để chủ đề này có điều kiện hoàn thiện hơn. Sau cùng, nhóm em kính chúc các thầy luôn dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ sau.

MỤC LỤC

CHƯƠNG I. GIỚI THIỆU

1. Mở đầu

Trong kỷ nguyên công nghệ số hiện nay, ngành Công nghệ thông tin (CNTT) đã đóng vai trò quan trọng trong việc phát triển các ứng dụng, phần mềm và trò chơi điện tử. Trong môn đồ án cơ sở ngành CNTT, chúng em đã chọn thực hiện dự án lập trình game Flappy Bird – một trò chơi điện tử được phát triển bởi tác giả Nguyễn Hà Đông vào năm 2013. Trò chơi nhanh chóng trở thành một hiện tượng toàn cầu với lối chơi đơn giản nhưng gây nghiện. Người chơi chỉ cần nhấn chuột hoặc chạm màn hình để làm cho nhân vật chính, một chú chim non, bay lên hoặc rơi xuống qua các ống nước treo đứng, đồng thời tránh va chạm với chúng. Mục tiêu của trò chơi là cố gắng bay càng xa càng tốt và đạt điểm số cao nhất. Đây là một trò chơi nhỏ dựa vào ngôn ngữ hướng đối tượng và các kiến thức về đồ họa.

2. Lý do chọn đề tài

Flappy Bird đã thu hút hàng triệu người chơi trên toàn thế giới nhờ lối chơi đơn giản nhưng gây nghiện. Việc phát triển phiên bản Flappy Bird bằng Java sẽ giúp chúng em tiếp cận và hiểu rõ hơn về cách tạo ra một trò chơi hấp dẫn. Bên cạnh đó, Java là một ngôn ngữ lập trình phổ biến và mạnh mẽ được sử dụng rộng rãi trong ngành công nghiệp game và phát triển ứng dụng. Qua đề tài này, chúng em hy vọng có thể nâng cao kỹ năng lập trình và khám phá thêm nhiều khả năng tùy chỉnh và mở rộng trò chơi. Chúng em tin rằng việc hoàn thành dự án này sẽ giúp chúng em tiếp thu kiến thức chuyên sâu về Java và phát triển sự sáng tạo trong lĩnh vực lập trình game.

3. Mục tiêu của đồ án

Mục tiêu chính của đồ án là xây dựng một phiên bản đơn giản của trò chơi Flappy Bird, đồng thời giải thích quy trình lập trình và các yếu tố thiết kế cơ bản trong quá trình phát triển. Chúng em muốn hiểu rõ hơn về cách hoạt động của trò chơi, cách xử lý va chạm, điều khiển nhân vật và cách tối ưu hóa hiệu suất của ứng dụng.

4. Ý nghĩa của đồ án

Đồ án cơ sở ngành Công nghệ thông tin – Lập trình game Flappy Bird đã cung cấp cho chúng em một cơ hội để:

- Học tập và áp dụng kiến thức CNTT.
- Hiểu rõ hơn về quá trình phát triển trò chơi.
- Nâng cao kỹ năng lập trình và sáng tạo.
- Áp dụng kiến thức vào thực tiễn.
- Nắm vững quy trình phát triển phần mềm.
- Được trải nghiệm cảm giác hoàn thành và có thành quả.

Tóm lại, ý nghĩa của đồ án lập trình game Flappy Bird trong môn đồ án cơ sở ngành CNTT là mang lại cơ hội học tập, áp dụng kiến thức, trải nghiệm quá trình phát triển phần mềm và cung cấp một sản phẩm hoàn thiện mà nhóm tác giả có thể hãnh diện.

5. Phạm vi nghiên cứu

Dự án tập trung vào việc xây dựng phiên bản cơ bản của trò chơi Flappy Bird dành cho máy tính cá nhân. Chúng em sẽ không mở rộng đến việc phát triển trên nền tảng di động hoặc tích hợp các tính năng phức tạp. Phạm vi đồ án giới hạn vào việc thực hiện các yêu cầu và chức năng cơ bản của trò chơi Flappy Bird.

6. Cấu trúc của đồ án

Phần tiếp theo của đồ án sẽ giới thiệu về lý thuyết và công nghệ liên quan đến lập trình game Flappy Bird. Sau đó, chúng em sẽ trình bày quá trình thiết kế và triển khai game, cùng với các thử nghiệm và đánh giá hiệu suất. Cuối cùng, chúng em sẽ tổng kết thành quả và trình bày những hướng phát triển tiếp theo.

Chúng em rất mong rằng đồ án này sẽ mang lại những kiến thức bổ ích và trải nghiệm thú vị, đồng thời đóng góp vào việc hiểu rõ hơn về lĩnh vực lập trình và ứng dụng CNTT trong lĩnh vực game điện tử.

CHƯƠNG II. CÔNG NGHỆ ỨNG DỤNG

1. Ngôn ngữ lập trình và môi trường phát triển

- Ngôn ngữ lập trình: Java – một trong những ngôn ngữ lập trình phổ biến, có thư viện và framework đa dạng hỗ trợ game và ứng dụng. Hơn nữa Java còn là một ngôn ngữ bảo mật đáng tin cậy có khả năng quản lý bộ nhớ tự động và nhiều lợi ích khác.
- Môi trường phát triển: IntelliJ IDEA Community Edition – một môi trường phát triển tích hợp (IDE) miễn phí dành cho ngôn ngữ lập trình Java cung cấp một loạt các tính năng và công cụ hỗ trợ lập trình viên trong việc phát triển các ứng dụng Java và Android.

2. Thư viện và giao diện người dùng

- Thư viện: Java.awt, javax.swing, java.io, java.imageio, javax.sound và java.util các thư viện này giúp người dùng dễ dàng thao tác với các thành phần giao diện, hình ảnh, âm thanh và dữ liệu trong việc phát triển game Flappy Bird cụ thể:
    - **java.awt (Abstract Window Toolkit):**

Là một bộ công cụ đồ họa trừu tượng, cung cấp các lớp và phương thức để xây dựng các giao diện đồ họa đơn giản trên Java. Các thành phần giao diện như cửa sổ, nút, hộp thoại, hình ảnh, v.v. được hỗ trợ trong thư viện này.
Dựa vào AWT, Java đã phát triển thư viện Swing để cung cấp giao diện người dùng phong phú và linh hoạt hơn.

- **javax.swing:**

Là một thư viện mở rộng từ AWT, cung cấp các thành phần giao diện người dùng phong phú và nâng cao hơn so với AWT. Swing hỗ trợ các thành phần giao diện như cửa sổ, nút, hộp thoại, menu, bảng, v.v., cùng với giao diện đồ họa trực quan và linh hoạt hơn so với AWT.

- **java.io:**

Là một gói dùng để thực hiện các hoạt động đọc và ghi dữ liệu. Hỗ trợ cho việc đọc và ghi dữ liệu từ các nguồn và đích khác nhau như tệp tin, ổ đĩa, đối tượng, v.v.

- **javax.imageio:**

Là một gói cung cấp các lớp và phương thức để thao tác với hình ảnh. Hỗ trợ đọc và ghi hình ảnh từ và vào các định dạng phổ biến như JPEG, PNG, GIF, v.v.

- **javax.sound:**

Là một gói cung cấp các lớp và phương thức để thao tác với âm thanh. Hỗ trợ phát và ghi âm thanh từ và vào các định dạng âm thanh khác nhau.

- **java.util:**

Là một gói chứa các lớp tiện ích dùng trong lập trình Java. Bao gồm các lớp hỗ trợ cho các cấu trúc dữ liệu như danh sách, tập hợp, bản đồ, hàng đợi, v.v. và các lớp tiện ích hỗ trợ cho các hoạt động như sắp xếp, tìm kiếm, định dạng, v.v.

3. Hệ thống kiến thức

Các kiến thức và kỹ năng cần nắm vững Khi lập trình game Flappy Bird bằng Java:

- Ngôn ngữ lập trình Java: Hiểu rõ cú pháp, cấu trúc, và cách hoạt động của Java để có thể viết mã nguồn cho game Flappy Bird.
- Xử lý sự kiện: Biết cách xử lý sự kiện như nhấn nút, di chuyển, và va chạm trong game để tạo ra sự tương tác giữa người chơi và trò chơi.
- Giao diện đồ họa: Sử dụng Java AWT hoặc Java Swing để tạo giao diện đồ họa cho game Flappy Bird, bao gồm cửa sổ, nút, hình ảnh, v.v.
- Vẽ và di chuyển đối tượng: Biết cách vẽ các đối tượng như chim và ống nước, và làm thế nào để di chuyển chúng trong không gian game.
- Xử lý va chạm: Hiểu cách phát hiện và xử lý va chạm giữa chim và ống nước, và quyết định khi nào game kết thúc khi xảy ra va chạm.
- Quản lý trạng thái game: Biết cách lưu và quản lý trạng thái của game, bao gồm trạng thái chơi, trạng thái kết thúc, và trạng thái chờ.
- Xử lý âm thanh: Biết cách sử dụng thư viện javax.sound để phát nhạc nền và các hiệu ứng âm thanh trong game.

- Gỡ lỗi và kiểm thử: Biết cách gỡ lỗi lỗi trong mã nguồn và kiểm thử game để đảm bảo tính đúng đắn và ổn định của game.
- Thiết kế giao diện người dùng: Để tạo giao diện người dùng dễ sử dụng và hấp dẫn trong game Flappy Bird.

CHƯƠNG III. SẢN PHẨM ĐỒ ÁN

1. Mô tả sản phẩm

Giao diện và điều khiển:

Giao diện của Flappy Bird rất đơn giản và đặc trưng bởi các đối tượng được vẽ 2D trên nền màu đơn giản. Người chơi điều khiển chim (hay con chim) bằng cách nhấn và giữ nút (hoặc nhấn và giữ màn hình trên các thiết bị cảm ứng) để thực hiện những cú đập cánh liên tục.

Mô tả cách chơi: Trò chơi Flappy Bird bắt đầu bằng việc đặt con chim ở trung tâm màn hình. Người chơi nhấn và giữ nút để làm con chim cất cánh, khi nút được thả ra, con chim rơi tự do xuống dưới. Nhiệm vụ của người chơi là vượt qua những ống nước (ống cống) được đặt ngang trên và dưới bằng cách di chuyển con chim qua giữa các khe hở của chúng. Người chơi nhận điểm bất kỳ khi con chim vượt qua một cặp ống nước, và mục tiêu là cố gắng ghi điểm cao nhất có thể.

Độ khó và thách thức: Trò chơi Flappy Bird được biết đến với độ khó cao và thách thức đòi hỏi người chơi có phản xạ nhanh và sự tập trung cao. Chim di chuyển rất nhanh, và người chơi phải điều khiển nhịp đập cánh sao cho đủ để vượt qua ống nước mà không va vào chúng hoặc chạm vào các cạnh màn hình.

Hiệu ứng âm thanh và hình ảnh: Flappy Bird có các hiệu ứng âm thanh đơn giản và hình ảnh đơn giản nhưng dễ nhận diện. Khi con chim va chạm vào ống nước hoặc cạnh màn hình, âm thanh va chạm sẽ được phát ra và con chim rơi xuống.

2. Yêu cầu và thiết kế

2.1 Yêu cầu:

Giao diện đơn giản và thân thiện: Flappy Bird nổi tiếng với giao diện đơn giản nhưng hấp dẫn. Cần có một giao diện thân thiện với người chơi, không quá rườm rà và dễ dàng để người chơi có thể tập trung vào việc chơi.

Đồ họa đơn giản: Thiết kế các hình ảnh, nhân vật và đối tượng trong trò chơi với phong cách đơn giản, dễ nhìn và dễ nhận diện. Cần có ít chi tiết và màu sắc tương phản để trò chơi dễ nhìn và tránh gây rối.

Tạo độ khó và thử thách: Đảm bảo trò chơi có mức độ khó và thử thách đủ để làm người chơi muốn cố gắng vượt qua điểm cao nhất. Các khoảng cách giữa ống nước và tốc độ di chuyển của con chim cần được cân nhắc để tạo ra sự hấp dẫn và gây nghiện.

Xử lý sự kiện và điều khiển chính xác: Phải đảm bảo các sự kiện nhấn và nhả nút hoặc cảm ứng màn hình diễn ra một cách chính xác và nhạy bén. Con chim cần phản ứng nhanh chóng khi người chơi bấm nút để thực hiện những cú đập cánh.

Âm thanh và hiệu ứng âm thanh phù hợp: Sử dụng âm thanh và hiệu ứng âm thanh phù hợp để tăng cường trải nghiệm chơi game. Ví dụ, âm thanh chim cất cánh, âm thanh va chạm khi con chim va vào ống nước hoặc cạnh màn hình.

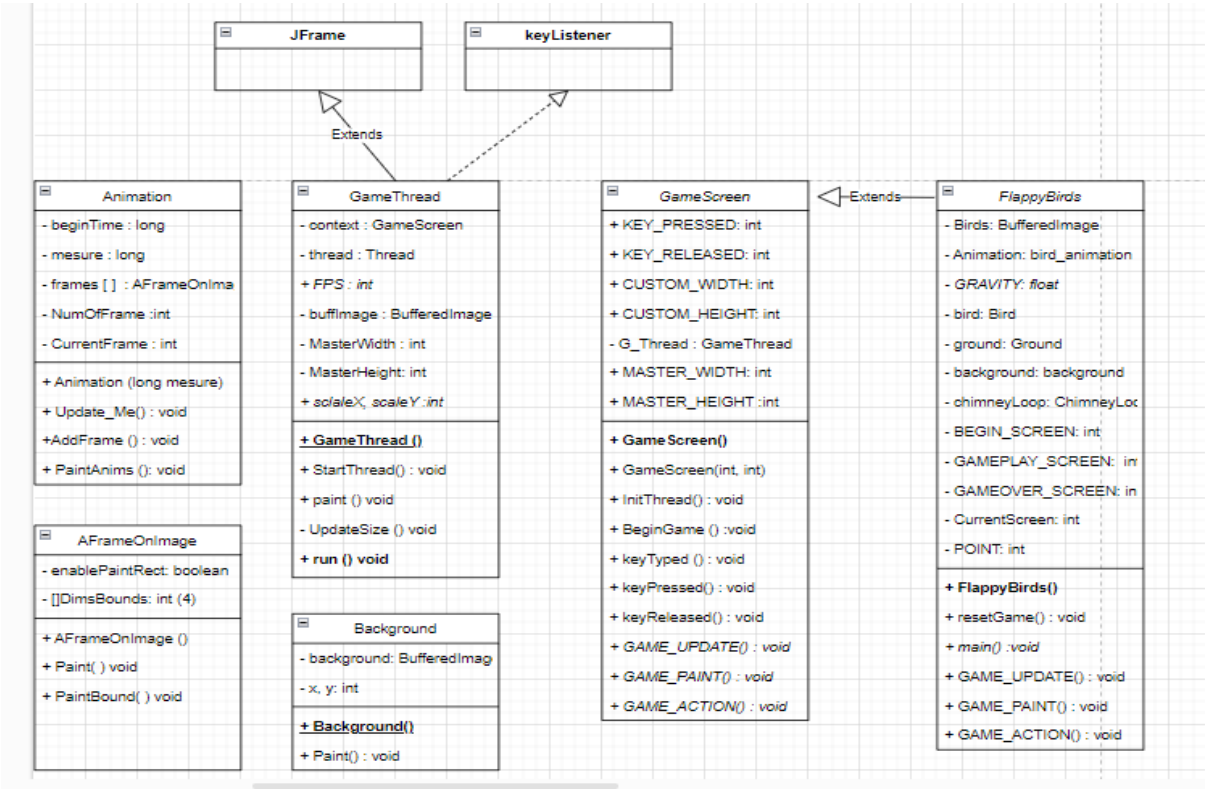
Gỡ lỗi và xử lý lỗi: Đảm bảo game được kiểm tra kỹ lưỡng để xác định và sửa các lỗi có thể xảy ra trong quá trình chơi. Tránh việc game bị treo hoặc gặp vấn đề kỹ thuật khi người chơi chơi.

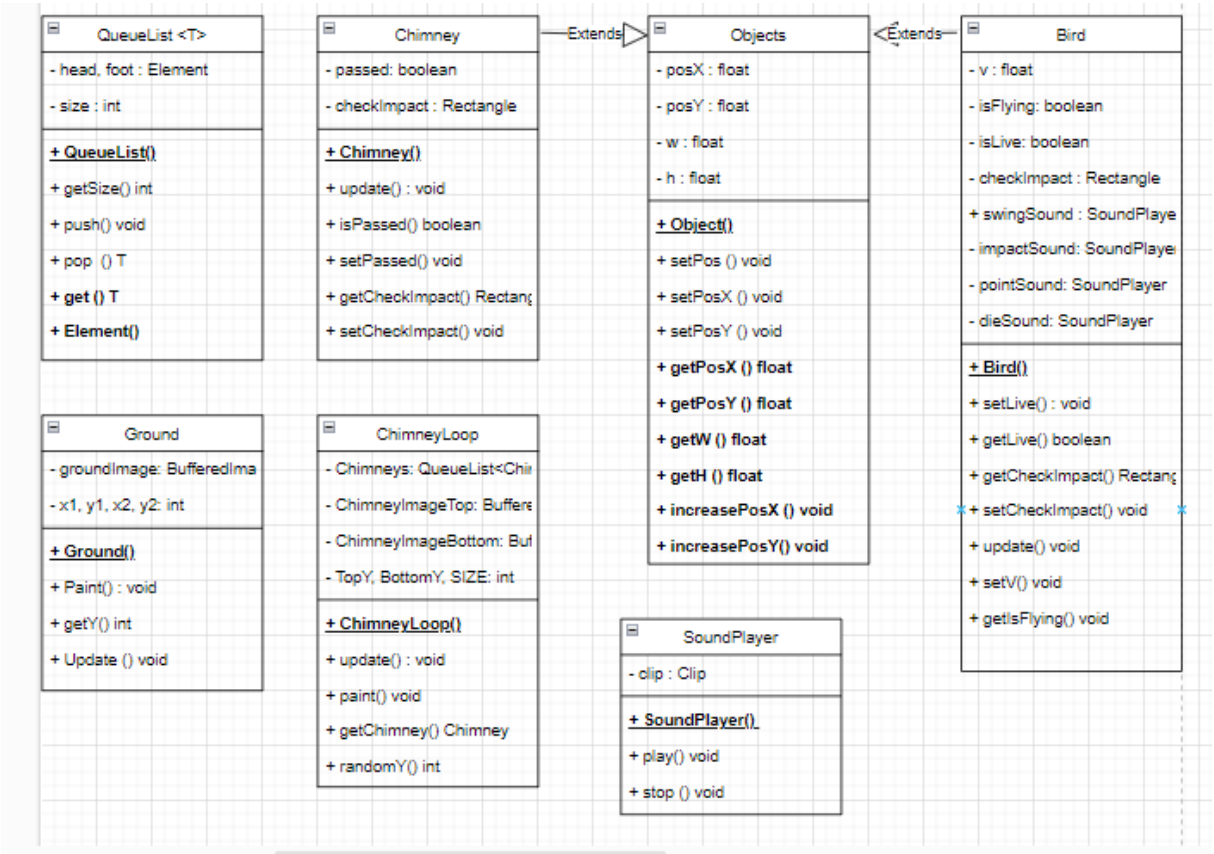
Lưu điểm số và thống kê: Cần cung cấp chức năng lưu điểm số của người chơi và hiển thị danh sách điểm cao nhất để khuyến khích sự cạnh tranh giữa người chơi.

2.2 Thiết kế

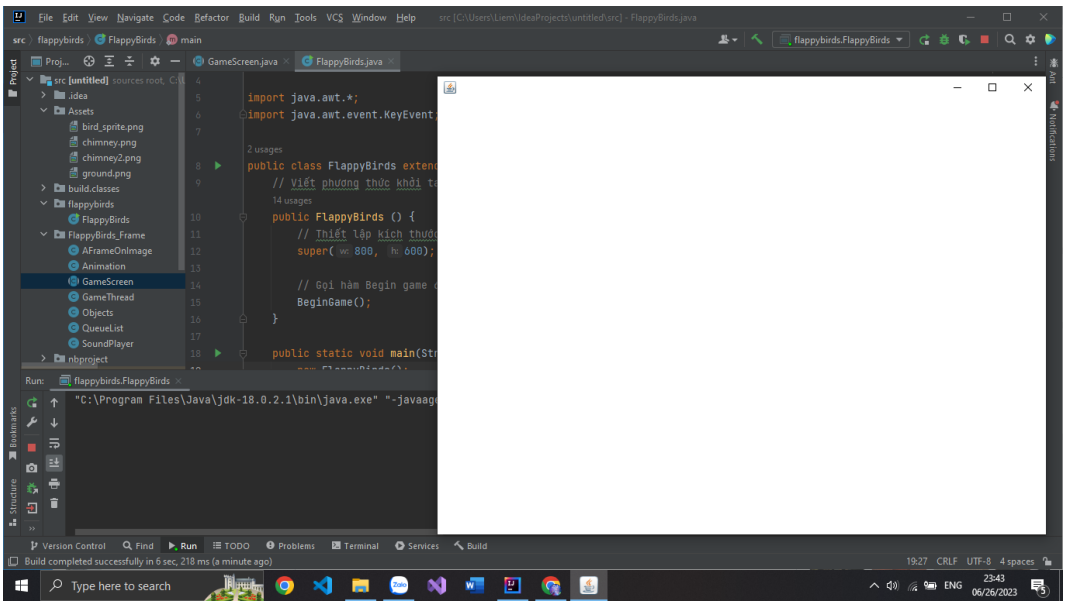
- **Thiết kế lớp (class)**

Thiết kế các class cơ sở của game như màn hình (GameScreen), chuyển động(Animation)… và các class đối tượng của trò chơi như Chim (Bird), ống khói (Chimney), mặt đất (Ground)…

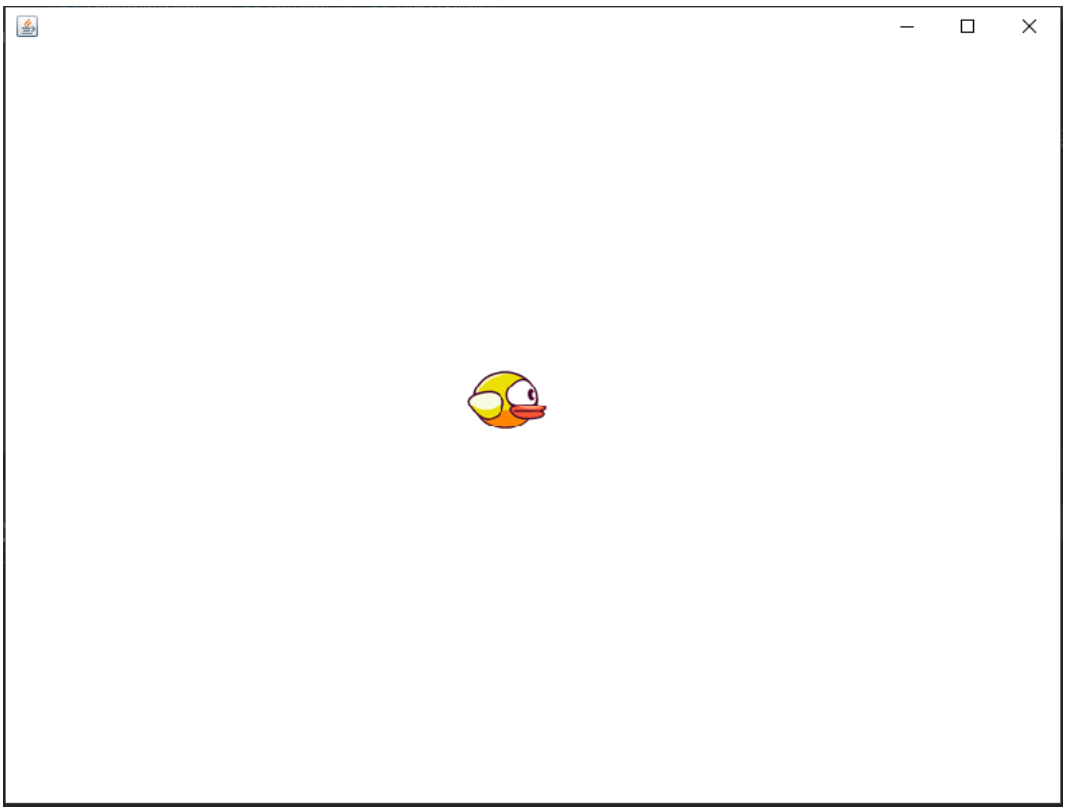




- **Tạo giao diện**



Tìm kiếm hình ảnh đồ họa của FlappyBirds và tạo một thư mục Asserts để thêm vào. Viết code cho phần chuyển động của đối tượng Bird và làm cho nó hoạt động được.



Lúc này dựa vào các frames ảnh và animation, hình ảnh Bird đã có thể chuyển động giữa các frames tạo hiệu ứng chim bay.

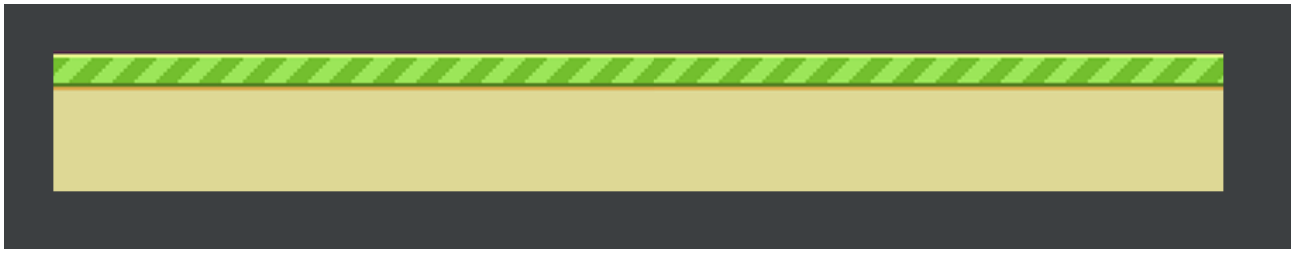
Tiếp theo sau đó, tạo thêm class Bird và thêm hiệu ứng rơi tự do và bay lên cho đối tượng Bird.

Trong đó đối tượng Bird sẽ có:

- Thuộc tính float v: thể hiện tốc độ rơi của chim.
- Phương thức update() để tăng tốc độ rơi của Bird bằng cách cộng dồn gia tốc hướng tâm GRAVITY .
- Phương thức fly() để giảm tốc độ v và tăng tọa độ của nó lên để tạo cảm giác bay lên, nhưng trong quá trình update thực hiện sẽ làm cho Bird rơi xuống.

- **Tạo đối tượng mặt đất**

- Hình ảnh:



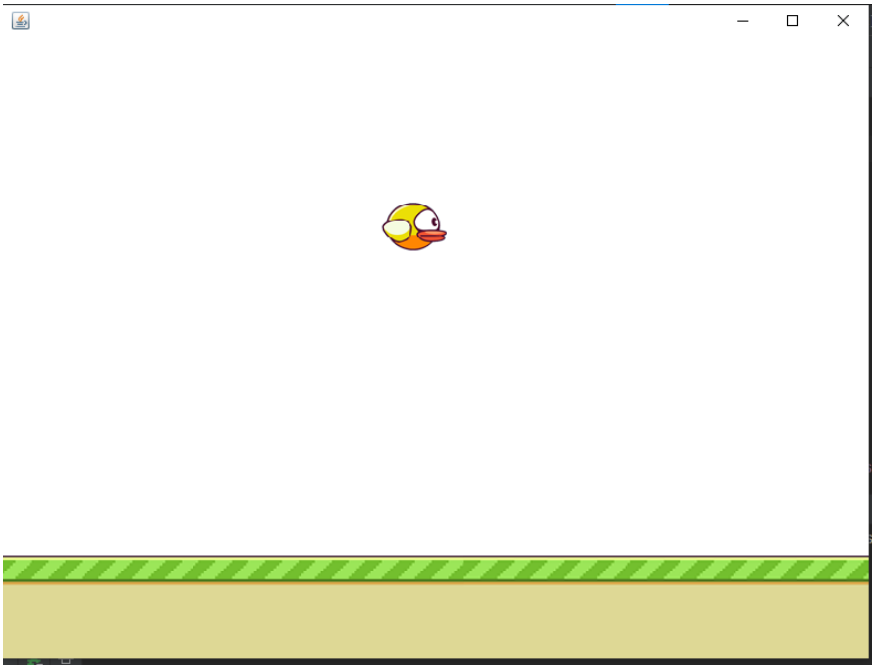
- Tạo class Ground

Ground
- groundImage(BufferedImage) - x (int) - y (int)
+ Ground() + Paint ()

Trong class FlappyBirds:

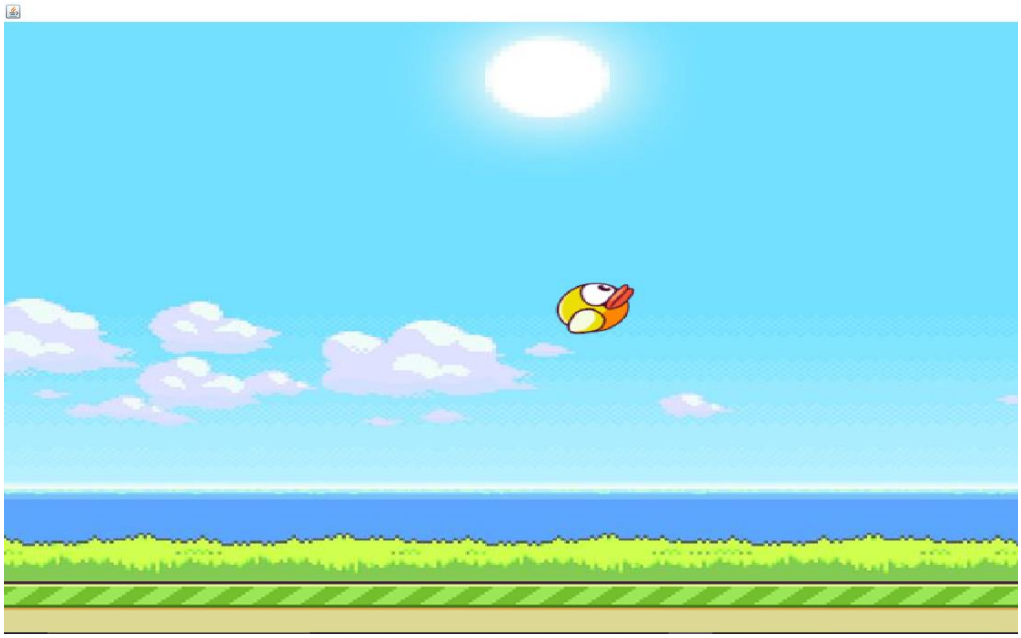
- Tạo biến – ground (Ground).
- Khởi tạo một Ground mới trong hàm khởi tạo FlappyBirds.
- Sau đó vào hàm GamePaint sử dụng ground.Paint() để vẽ một mặt đất trong màn hình hiển thị.

- Kết quả:



Tạo tính di chuyển cho mặt đất: Mặt đất khi khởi tạo ở bước 1 chưa có tính di chuyển, để tạo tính di chuyển ta cần update tọa độ cho mặt đất đầu tiên di chuyển, tuy nhiên khi di chuyển hết chiều dài bức ảnh mặt đất thì sẽ lộ ra khoảng trắng. Để chỉnh sửa điều này, cần tạo ra một mặt đất phía sau(2) mặt đất đầu tiên(1) và cho nó thuộc tính di chuyển tương tự như mặt đất đầu tiên, nhưng cần thay đổi vị trí của mặt đất số 1 ra phía sau mặt đất số 2 khi mặt đất số 2 chạm trục y, và tương tự như thế cho mặt đất số 2.

- **Tạo background cho trò chơi**



Tương tự như phần tạo mặt đất, nhưng không có phần di chuyển background (nếu có vẫn xây dựng được).

```
2 usages
public class Background {
    2 usages
    private int x, y; // Tọa độ của background
    2 usages
    private BufferedImage background;
    1 usage
    public Background () {
        try {
            background = ImageIO.read(new File( pathname: "Assets/background.jpg"));
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        x = -50; y = -50;
    }

    1 usage
    public void Paint (Graphics2D g2) {
        g2.drawImage(background, x, y, observer: null);
    }
}
```

Lưu ý: khi vẽ các đối tượng, các đối tượng nào được vẽ trước thì sẽ hiện ở phía dưới màn hình, dưới các đối tượng khác. Các đối tượng nào vẽ sau sẽ hiện ở phía trên các đối tượng khác.

- **Viết code chia màn hình game**

Chia màn hình game thành 3 màn hình chính gồm:

- Màn hình bắt đầu.

- Mình hình chơi game.
- Màn hình kết thúc.

Thực hiện như sau:

```
6 usages
private int BEGIN_SCREEN = 0;
3 usages
private int GAMEPLAY_SCREEN = 1;
4 usages
private int GAMEOVER_SCREEN = 2;
12 usages
private int CurrentScreen = BEGIN_SCREEN;
```

Chia phần ở các hàm được Override để phân biệt màn hình khi nào chơi game, khi nào thực hiện update và khi nào kết thúc.

```
1 usage
@Override
public void GAME_UPDATE(long deltaTime) {
    if(CurrentScreen == BEGIN_SCREEN) {

    } else if (CurrentScreen == GAMEPLAY_SCREEN) {
        // Khi game được bắt đầu chơi mới thực hiện update
        bird_animation.Update_Me(deltaTime);
        bird.update(deltaTime);
        ground.Update();

        // Kiểm tra khi nào bị va chạm và kết thúc game đối với mặt đất
        if (bird.getPosY() + bird.getH() > ground.getY()) CurrentScreen = GAMEOVER_SCREEN;
    } else  if (CurrentScreen == GAMEOVER_SCREEN){
        CurrentScreen = BEGIN_SCREEN;
    }
```

```
1 usage
@Override
public void GAME_PAINT(Graphics2D g2) {
    background.Paint(g2);
    ground.Paint(g2);

    if(bird.getIsFlying()) // trả về true => đang bay lên
        bird_animation.PaintAnims((int) bird.getPosX(), (int) bird.getPosY(), Birds, g2,  anchor: 0,  rotation: -1 );
    else
        bird_animation.PaintAnims((int) bird.getPosX(), (int) bird.getPosY(), Birds, g2,  anchor: 0,  rotation: 0 );
    if(CurrentScreen == BEGIN_SCREEN) {
        g2.setColor(Color.black);
        g2.drawString( str: "Press SPACE to play game",  x: 310,  y: 240);
    }
    if (CurrentScreen == GAMEOVER_SCREEN) {
        g2.setColor(Color.black);
        g2.drawString( str: "Press SPACE to play game again!",  x: 310,  y: 240);
    }
}
```

```
2 usages
@Override
public void KEY_ACTION(KeyEvent e, int Event) {
    if(CurrentScreen == BEGIN_SCREEN) {
        CurrentScreen = GAMEPLAY_SCREEN;
    } else if (CurrentScreen == GAMEPLAY_SCREEN) {
        bird.fly(); //if (Event == KEY_PRESSED) bird.fly();
    } else if (CurrentScreen == GAMEOVER_SCREEN){
        CurrentScreen = BEGIN_SCREEN;

    }
}
```

Tạo phương thức reset game khi Bird chạm vào mặt đất:

- Tạo hàm set V () ở class Bird.
- Tạo hàm resetGame() ở class FlappyBird: khởi tạo lại vị trí ban đầu và tốc độ của Bird.

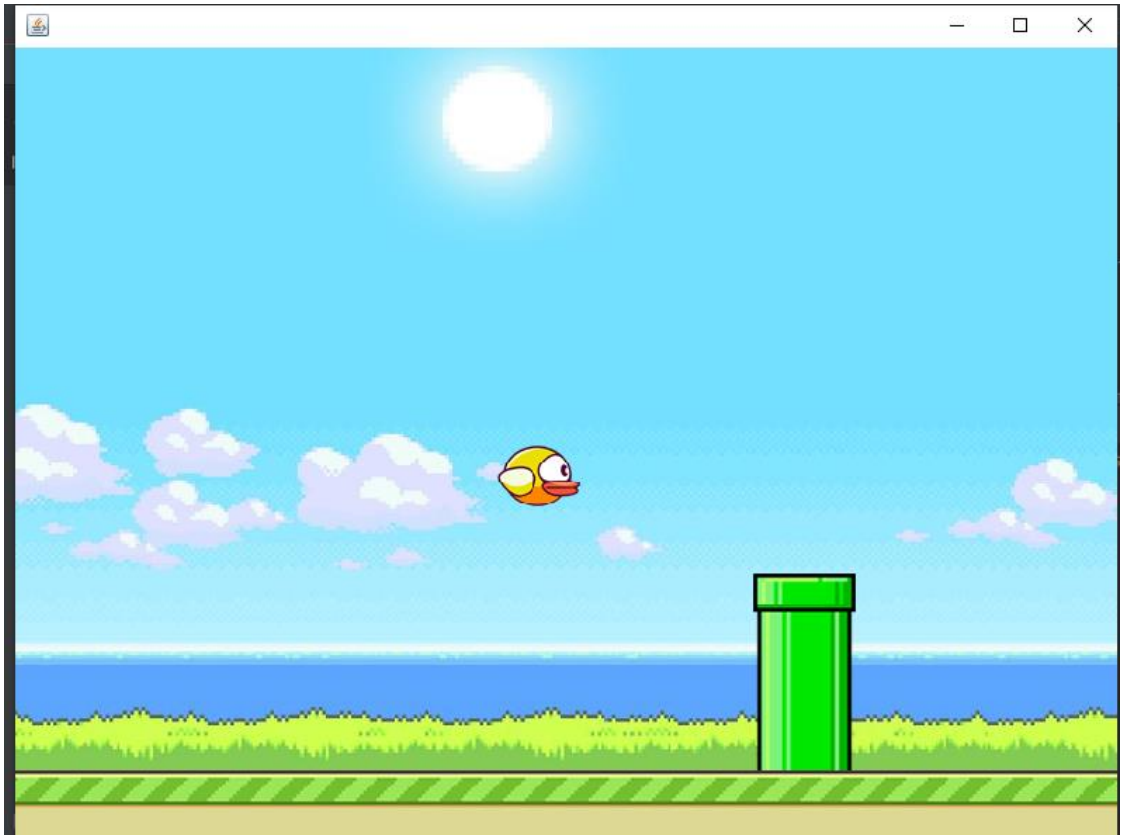
- **Viết code tạo ống khói**

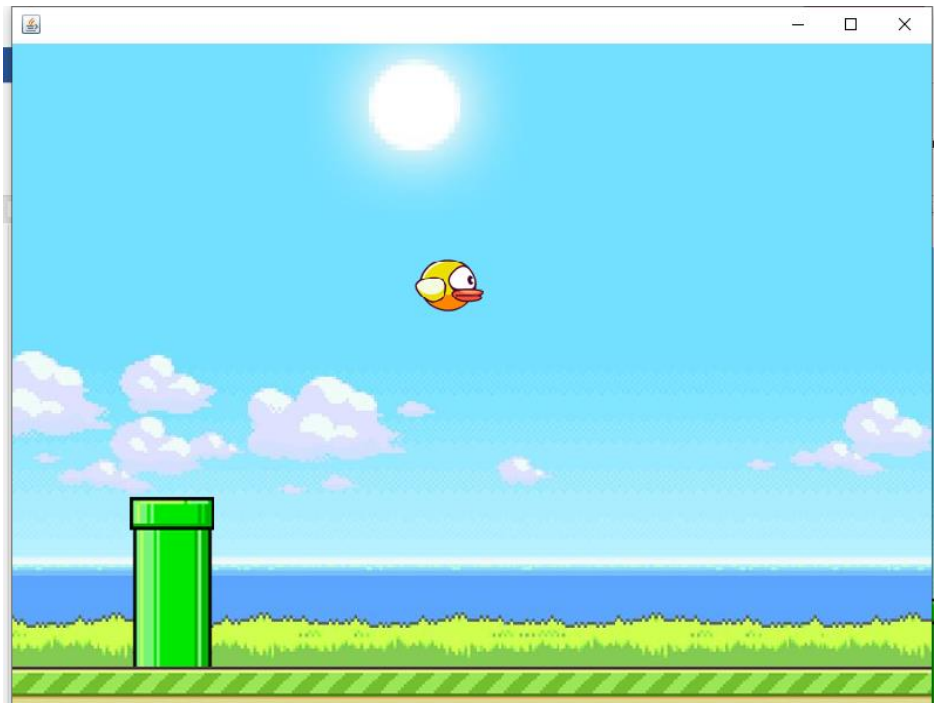
- Đầu tiên, tạo một class Chimney kế thừa từ lớp Objects.
- Tạo phương thức khởi tạo và phương thức update

```
2 usages
public class Chimney extends Objects {
    1 usage
    public Chimney (int x, int y, int width, int height) {
        super(x, y, width, height);

    }

    1 usage
    public void update() {
        setPosX(getPosX()-2);
    }


}
```

- Phương thức khởi tạo nhằm kế thừa phương thức khởi tạo từ lớp Objects để xác định vị trí của đối tượng và chiều rộng chiều cao của đối tượng được vẽ.
- Phương thức update sử dụng để tạo sự di chuyển của ông khói theo trục Ox.

- Ở class Flappy Bird, ta khởi tạo hai biến private BufferedImage Chimneys và một biến private Chimney chimney để tạo đối tượng từ class Chimney.
- Sau đó sử dụng ImageIO.read(new File()) để đọc lấy bức ảnh ống khói.
- Vẽ ống khói ở hàm GAME_PAINT và tạo chuyển động của ống khói ở hàm GAME_UPDATE.





- **Viết code tạo vòng lặp ống khói:** Tạo một class mới ChimneyLoop.
  - Xóa hình ảnh ống khói đơn ở Class FlappyBird.
  - Trong class này, tạo 2 bức ảnh 2 ống khói phía trên và phía dưới.
  - Tạo một Queuelist các đối tượng của class Chimney.
  - Sau đó đọc file hình ảnh để lấy ra hình ảnh của ống khói.
  - Sử dụng vòng lặp để tạo ống khói ở dưới và trên của các cặp ống khói.

```
1 usage
public ChimneyLoop() {
    Chimneys = new QueueList<Chimney>();
    Chimney chimney;

    // Lấy một hình ảnh Flappy Birds bằng ImageIO
    try {
        ChimneyImageTop = ImageIO.read(new File( pathname: "Assets/chimneytop.png"));
        ChimneyImageBottom = ImageIO.read(new File( pathname: "Assets/chimneybuttom.png"));
    } catch (IOException e) {
        throw new RuntimeException(e);
    }

    // Tạo 3 cặp ống khói
    for (int i = 0; i < 3; i++) {
        // Ống khói thứ nhất
        chimney = new Chimney( x: 830 + i * 300, y: 350, width: 74, height: 400);
        Chimneys.push(chimney);

        // Ống khói thứ hai
        chimney = new Chimney( x: 830 + i * 300, y: -300, width: 74, height: 400);
        Chimneys.push(chimney);
    }
}
1 usage
```

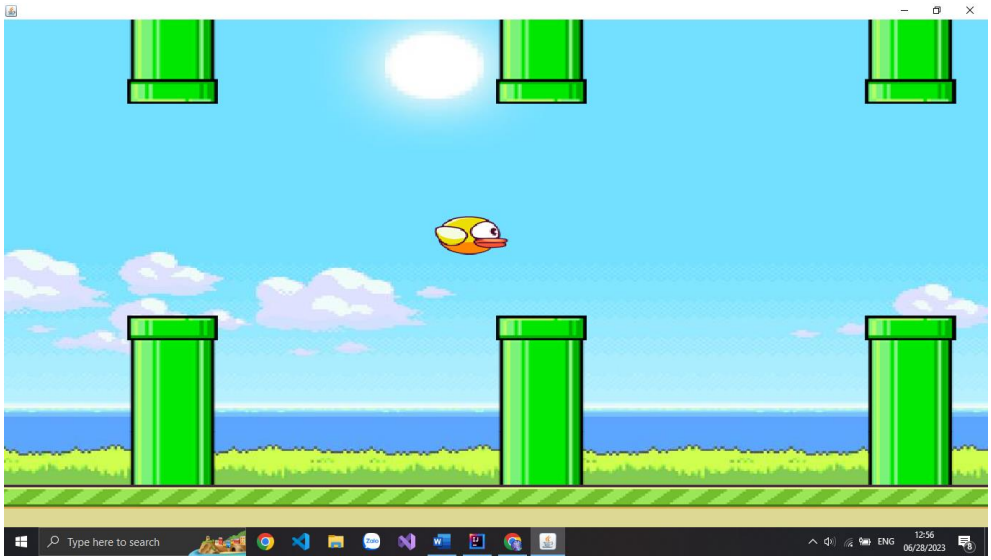
- Viết hàm update – gọi vòng for để với mỗi đối tượng ống khói trong queuelist. Ta gọi hàm update của nó từ class Chimney đã xây dựng sẵn, nhằm tạo hiệu ứng di chuyển.
- Và nếu mỗi ống khói di chuyển đến vị trí âm – tức là đã bị ẩn sau màn hình thì mình sẽ sử dụng pop để xóa nó ra khỏi queuelist và update lại tọa độ ban đầu phía sau ống khói cuối cùng, rồi sau đó lại push vào trong queuelist để đối tượng có thể được vẽ ra.

```
1 usage
public void update() {
    for (int i = 0; i < 6; i++) {
          Chimneys.get(i).setPosX(Chimneys.get(i).getPosX() - 2);
        // Update tọa độ của ôngống khói, tạo sự di chuyển
        Chimneys.get(i).update();

    }
    // Nếu tọa độ của ống khói < -74 thì xóa nó ra khỏi queue list và set tọa độ
    // nó ở phía sau tạo độ của ôống khói cuối cùng
    if(Chimneys.get(0).getPosX() < -74 )
    {
        Chimney chimney;
        chimney = Chimneys.pop();
        chimney.setPosX(Chimneys.get(4).getPosX() + 300);
        Chimneys.push(chimney);

        chimney = Chimneys.pop();
        chimney.setPosX(Chimneys.get(4).getPosX() );
        Chimneys.push(chimney);
    }
}
```

- Có tổng cộng là 3 cặp ống khói, vì vậy sử dụng vòng lặp for(6) để vẽ mỗi ống khói trên và dưới của mỗi cặp

```
1 usage
public void paint (Graphics2D g2) {
    for (int i = 0; i < 6; i++) {
        if (i % 2 == 0) // Nếu là ống khói thứ nhất thì vẽ nó quay lên
            g2.drawImage(ChimneyImageBottom, (int) Chimneys.get(i).getPosX(), (int) Chimneys.get(i).getPosY(), observer: null);
        else // Ngược lại vẽ ống khói thứ hai quay xuống
            g2.drawImage(ChimneyImageTop, (int) Chimneys.get(i).getPosX(), (int) Chimneys.get(i).getPosY(), observer: null);

    }
```

Sau đó viết lại các hàm vẽ ống khói có trong class chính Flappy Bird. Ta được kết quả như sau:

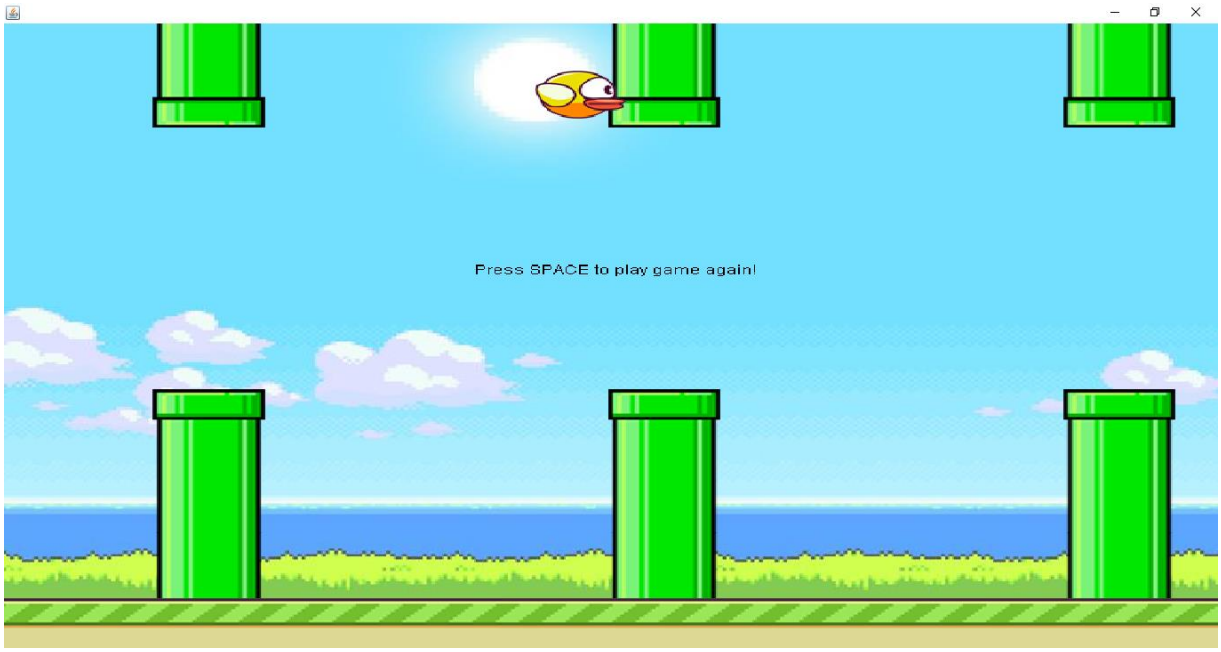


- **Viết code tạo tính năng va chạm giữa Bird và Chimney thì game sẽ dừng lại.**
  - Ở class Chimney và Bird, ta tạo thêm thuộc tính – checkImpact(Rectangle) và phương thức get, set
  - Cập nhật ở hàm khởi tạo checkImpact = new Rectangle để thêm tạo độ đối tượng vào
  - Cập nhật ở hàm update checkImpact.setLocation để cập nhật vị trí của đối tượng khi di chuyển.

```
8 usages
public class Chimney extends Objects {
    4 usages
    private Rectangle checkImpact;

    1 usage
    public Rectangle getCheckImpact() {
        return checkImpact;
    }

    public void setCheckImpact(Rectangle checkImpact) {
        this.checkImpact = checkImpact;
    }

    2 usages
    public Chimney (int x, int y, int width, int height) {
        super(x, y, width, height);
        checkImpact = new Rectangle(x, y, width, height);

    }

    1 usage
    public void update() {
        setPosX(getPosX()-2);
        this.checkImpact.setLocation((int) this.getPosX(), (int) this.getPosY());

    }
```

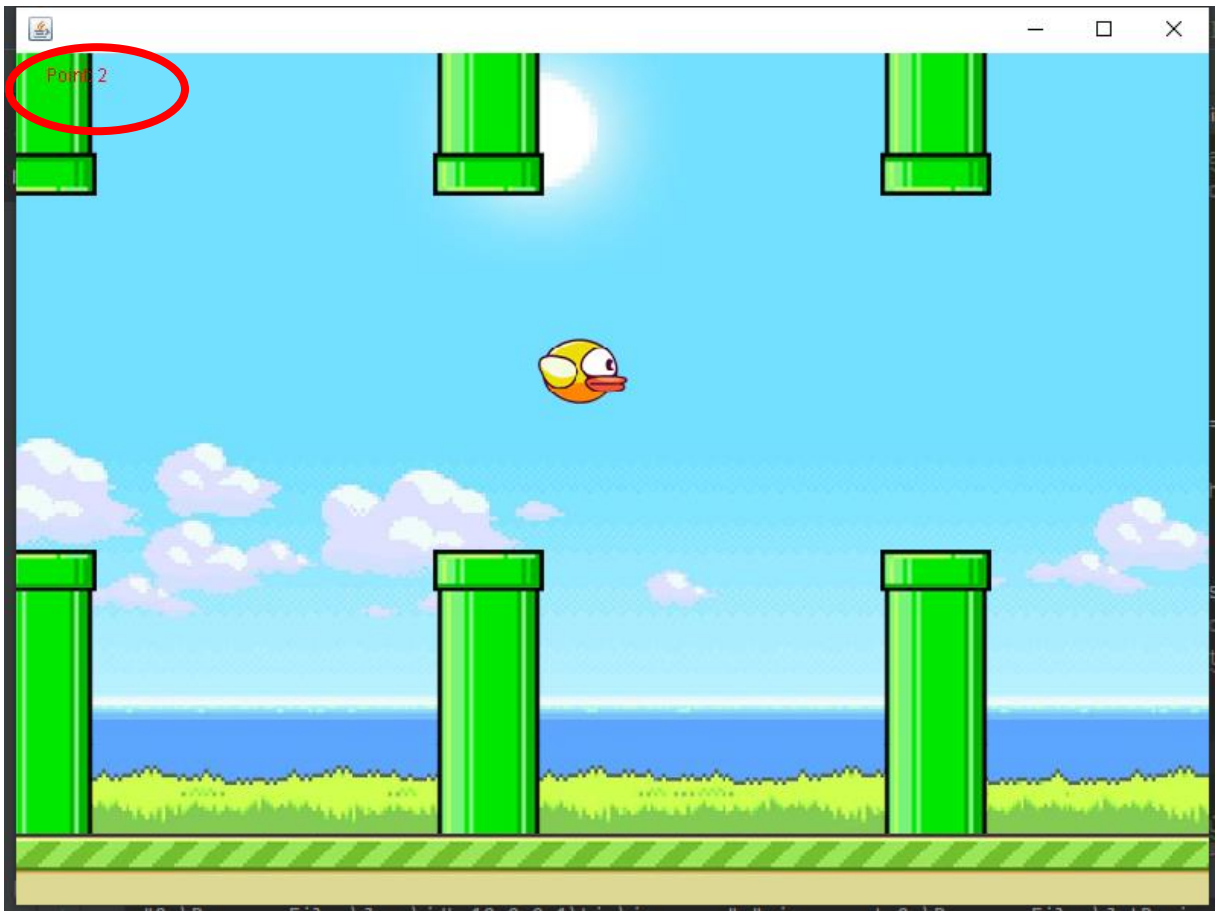
```java
2 usages
private boolean isLive = true;

1 usage
public Rectangle getCheckImpact() {
    return checkImpact;
}

public void setCheckImpact(Rectangle checkImpact) {
    this.checkImpact = checkImpact;
}

// biến kiểm tra va chạm giữa Bird và Chimney
// bằng cách kiểm tra va chạm giữa các hình chữ nhật với nhau
4 usages
private Rectangle checkImpact;
1 usage
public Bird (int x, int y, int width, int height) {
    super(x, y, width, height);
        // Trong đó x và y là tọa độ của con chim và width - height là độ
        // rộng của con chim để xác định va chạm với vùng khác

    checkImpact = new Rectangle(x, y, width, height);

}
```

- Ở class Bird, tạo một thuộc tính isLive của Bird ta kiểm tra trạng thái của Bird khi game đang chơi ở màn hình Flappy Bird như sau:
    - Nếu đối tượng Bird giao với đối tượng của Chimney thông qua hàm intersects thì ta sẽ cho trạng thái của Bird là false.
    - Khi trạng thái của Bird là true thì các hàm update hoạt động bình thường, ngược lại ngừng hoạt động.
    - Khi trạng thái của Bird là false thì CurrentScreen lúc này là GAMEOVER_SCREEN.
    - Reset game, trạng thái của Bird là true.



- **Viết code tạo điểm số cho trò chơi**
  - Ở mỗi ống khói, ta thêm một thuộc tính boolean là Passed = false, nếu Bird đã đi qua ống khói rồi thì set Passed = true và tăng điểm. Vì vậy, ở class Chimney, tạo thêm một thuộc tính Passed = false, khởi tạo phương thức get, set.
  - Vấn đề là khi 3 cặp ống khói đã đi qua hết và tất cả thuộc tính Passed đã là true, chúng ta cần set lại Passed là false ở hàm update tất cả các ống khói.
  - Ở class Flappy Bird, ta cần kiểm tra xem với mỗi cặp ống khói, nếu tọa độ của Bird hiện tại lớn hơn tọa độ của ống khói + độ dày của ống khói (74) thì biến POINT lưu điểm sẽ tăng lên 1, và lúc này Trạng thái của ống khói Passed là true.

```
for (int i = 0; i < ChimneyLoop.SIZE; i+=2) {
    // Tính điểm bằng cách kiểm tra
    if(bird.getPosX() > chimneyLoop.getChimney(i).getPosX() + 74  && !chimneyLoop.getChimney(i).isPassed())
    {
        POINT++;
        chimneyLoop.getChimney(i).setPassed(true);
        // Khi đã qua ó ống khói, không còn cái nào flase nữa,
        // nên ta phải update lại trạng thái của ống khói khi nó về sau
    }
}
```



- **Tạo lại game khi game over**
  - Về tính chất, thì Bird đứng yên và các ống khói, mặt đất di chuyển, nên khi game over và người dùng muốn chơi lại, ta chỉ cần gọi lại hàm khởi tạo của ChimneyLoop để nó reset tọa độ của các ống khói và đồng thời reset lại trạng thái ban đầu của Bird khi ở màn hình game bắt đầu.

```
1 usage
public void resetGame () {
    bird.setPos( x: 350,  y: 250);
    bird.setV(0);
    bird.setLive(true);
    POINT = 0;
    // Khởi tạo lại màn hình game khi Game Over
    chimneyLoop = new ChimneyLoop();
}
```

- **Tạo vị trí ngẫu nhiên cho các ống khói (thay vì thẳng hằng như trước đến giờ)**
  - Ta nhận thấy rằng vị trí của ống khói được vẽ chênh lệnh nhau dựa trên tọa độ của điểm y, vì vậy ta có thể sử dụng hàm random để lấy một giá trị int nào

đó cho biến y và sau đó cộng nó vào tọa độ của cặp ống khói thì ta sẽ có được tọa độ ngẫu nhiên của ống khói.

- Thực hiện:
  - Tạo vị trí cho ống khói để cố định khoảng cách giữa 2 ống khói. (Top = - 350, Bottom = 200)

```
2 usages
private int TopY = -350, BottomY = 200;
```

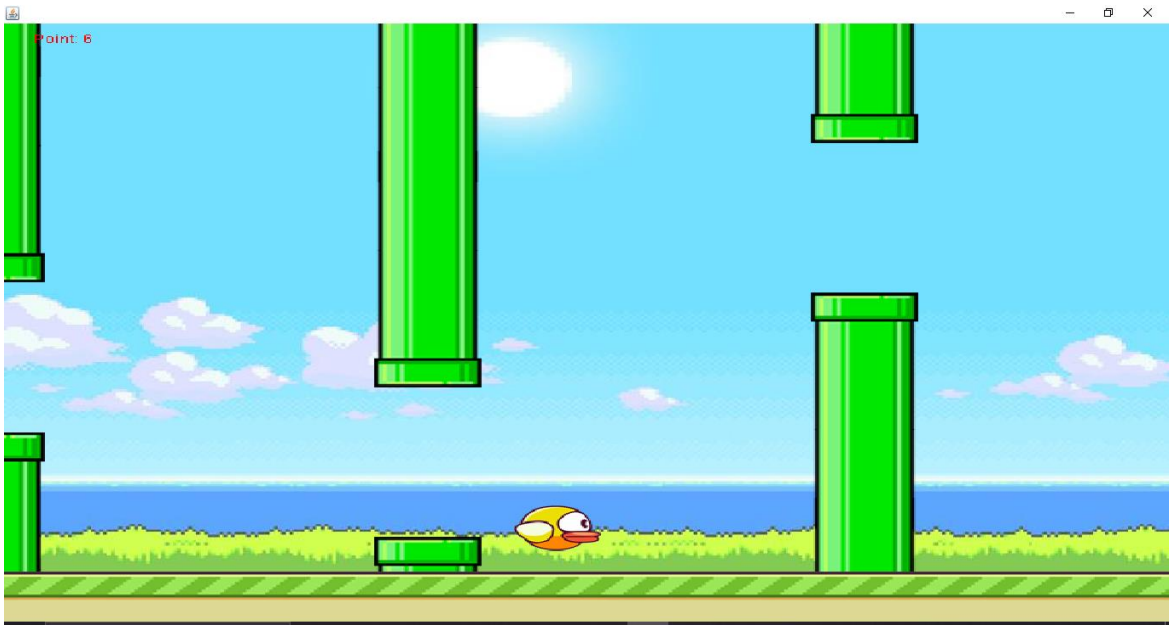
  - Tạo hàm random theo cấp (chiều dài của ống khói là 400 , chia thành 10 cấp, và cho khoảng cách mỗi cấp cách nhau một số random * 35 (trừ cho 2 mép trên dưới))

```
2 usages
public int randomY () {
    Random random = new Random();
    int Y = random.nextInt( bound: 10);
    return Y * 35;
}
```

  - Update tạo độ khi vẽ các ống khói lần đầu và khi reset lại tọa độ

```
// Tạo 3 cặp ống khói
for (int i = 0; i < SIZE / 2; i++) {
    int Y = randomY();
    // Ống khói thứ nhất
    chimney = new Chimney( x: 830 + i * 300,  y: BottomY + Y,  width: 74,  height: 400);
    Chimneys.push(chimney);

    // Ống khói thứ hai
    chimney = new Chimney( x: 830 + i * 300,  y: TopY + Y,  width: 74,  height: 400);
    Chimneys.push(chimney);
}
```

```
// Nếu tọa độ của ống khói < -74 thì xóa nó ra khỏi queue list và set tọa độ
// nó ở phía sau tạo độ của ôống khói cuối cùng
if(Chimneys.get(0).getPosX() < -74 )
{   int Y = randomY();
    Chimney chimney;
    chimney = Chimneys.pop();
    chimney.setPosX(Chimneys.get(4).getPosX() + 300);
    chimney.setPosY(BottomY + Y);
    Chimneys.push(chimney);
    chimney.setPassed(false);


    chimney = Chimneys.pop();
    chimney.setPosX(Chimneys.get(4).getPosX() );
    chimney.setPosY(TopY + Y);
    Chimneys.push(chimney);
    chimney.setPassed(false);

}
```



- **Tạo âm thanh cho game:** Tương tự như hình ảnh, tại class Bird, khởi tạo các thuộc tính sound cho các hành dộng va chạm, bay lên, cộng điểm và rơi xuống đất.

```
2 usages
public SoundPlayer swingSound, impactSound, pointSound, dieSound;
```

Ở đây ta khởi tạo thuộc tính là public để các lớp khác có thể sử dụng được, đặc biệt là lớp Flappy Bird.

Tương tự hình ảnh, ta đặt các biến vào trong hàm khởi tạo của class Bird và chọn đường dẫn âm thanh thích hợp.

```
1 usage
public Bird (int x, int y, int width, int height) {
    super(x, y, width, height);
        // Trong đó x và y là tọa độ của con chim và width - height là độ
        // rộng của con chim để xác định va chạm với vùng khác

    checkImpact = new Rectangle(x, y, width, height);
    swingSound = new SoundPlayer(new File( pathname: "Assets/wing.wav"));
    impactSound = new SoundPlayer(new File( pathname: "Assets/hit.wav"));
    pointSound = new SoundPlayer(new File( pathname: "Assets/point.wav"));
    dieSound = new SoundPlayer(new File( pathname: "Assets/die.wav"));



}
```

Sau đó, với mỗi phương thức kiểm tra, ta sẽ thêm âm thanh phù hợp vào từng phương thức để tạo âm thanh cho game.

Ví dụ :

```
1 usage
public void fly () {
    v = (float) - 2.5; // Lúc này v sẽ tăng lên
    // Nhưng trong quá trình update nó sẽ lại tăng giá trị v này nên => giảm theo y rơi xuống lại
    swingSound.play();
}
```

Như vậy đã tạo xong game có âm thanh, bước tiếp theo sẽ làm điểm số cao nhất và tìm cách chuyển màn hình khi chơi đến một điểm số nhất định cũng như tăng độ khó của game khi tăng tốc độ di chuyển và số lượng ống khói.

3. Ưu điểm và hạn chế

- **Ưu điểm:**
    - Đơn giản và dễ chơi
    - Thời gian chơi ngắn
    - Phù hợp với nhiều đối tượng
    - Có sức lan tỏa nhanh chóng
- **Nhược điểm:**
    - Độ khó và thử thách cao

- Thiếu đa dạng
- Giao diện đồ họa đơn giản
- Dễ gây nhàm chán

4. Ứng dụng

Là một tựa game giải trí kinh điển, Flappy Bird có thể được sử dụng như một ví dụ minh họa hoặc một bài học trong việc lập trình game, đặc biệt là đối với những người mới bắt đầu hoặc những người muốn học về lập trình game. Từ đó giúp người học có thể hiểu rõ hơn về cơ chế chơi, xử lý sự kiện, va chạm và quản lý dữ liệu trong game.

CHƯƠNG IV. KẾT LUẬN

Tóm lại, dựa vào kiến thức lập trình hướng đối tượng Java đã có, kết hợp với kiến thức của môn học đồ họa máy tính tạo ra giao diện các thực thể tương tác với nhau và có thể hoạt động như một game đồ họa 2D. Đồ án cơ sở ngành – chủ đề lập trình game Flappy Bird đã giúp cho sinh viên nắm vững kiến thức về lập trình hướng đối tượng sử dụng ngôn ngữ Java và kết hợp với kiến thức đồ họa máy tính học được để tạo ra một sản phẩm hoàn chỉnh. Hơn nữa đồ án cũng giúp cho nhóm em bắt đầu làm quen với lập trình game, hiểu rõ hơn về cơ chế chơi của một game đơn giản, xử lý các sự kiện, va chạm và quản lý dữ liệu trong game.

TÀI LIỆU THAM KHẢO

- Thông tin về game Flappy Birds: https://vi.wikipedia.org/wiki/Flappy_Bird
- Hướng dẫn lập trình trò chơi Java – Flappy Bird redux: https://www.instructables.com/Java-Game-Programming-Tutorial-Flappy-Bird-Redux/
- Lập trình game Flappy Bird trên Java: https://www.youtube.com/watch?v=GjJfRi3qpg4