ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ

ĐỒ ÁN MÔN LẬP TRÌNH TRỰC QUAN

Giảng viên hướng dẫn:

Ths. TRẦN ANH DŨNG

Sinh viên thực hiện:

TRẦN QUỐC THẮNG	**19522218**
VŨ NGỌC MỸ PHƯƠNG	**19522071**
PHẠM NGỌC QUYÊN	**19522115**
VÕ TRUNG TÍN	**19522353**

Tp. Hồ Chí Minh, 1/2021

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

....., ngày…….tháng……năm 2020

Người nhận xét

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trước khi đi vào nội dung phần báo cáo đồ án, lời nói đầu tiên chúng em xin chân thành gửi lời cảm ơn đến **Ths. TRẦN ANH DŨNG** – Giảng viên môn **Lập Trình Trực Quan** – người đã nhiệt tình giảng dạy trên lớp, cung cấp cho chúng em rất nhiều kiến thức hữu ích trong quá trình mới bắt đầu làm quen với ngôn ngữ C# và cách thức lập trình trên Windows Form, đồng thời thầy cũng hỗ trợ những thông tin cần thiết, gợi mở những trường hợp sẽ gặp phải vấn đề khi lập trình trong thực tế, đưa ra những giải pháp tối ưu và giải đáp những thắc mắc không chỉ cho nhóm chúng em nói riêng và các bạn sinh viên khác trên lớp nói chung trong suốt quá trình học tập và thực hiện đồ án.

Đồng thời, nhóm chúng em cũng muốn gửi lời cảm ơn sâu sắc đến những người đã hỗ trợ chúng em trong quá trình thực hiện đồ án này, không chỉ là những người bạn đã tạo điều kiện thuận lợi và động viên chúng em rất nhiều, mà đặc biệt là những anh chị trong khoa Công Nghệ Phần Mềm vì đã góp ý, chia sẻ những kinh nghiệm quý báu về môn học cũng như các kiến thức khác có liên quan.

Nhóm thực hiện

Trường Đại học Công nghệ Thông tin, tháng 1 năm 2021

LỜI MỞ ĐẦU

Trong bối cảnh khoa học và kĩ thuật theo xu hướng 5.0 ngày nay, chất lượng cuộc sống con người không ngừng được cải thiện và nâng cao, từ việc ngày càng có nhiều dịch vụ, các ứng dụng tiện ích, các thiết bị thông minh, … đáp ứng nhu cầu giải trí của con người cho đến khối lượng công việc được cắt giảm bởi sự tham gia, hợp thức hóa trong các hoạt động sinh hoạt và sản xuất của các máy móc, robot thông minh. Riêng tại Việt Nam, các trò chơi, ứng dụng giải trí trên các máy tính, thiết bị di động đang ngày càng thu hút được nhiều người dùng, việc này được xem là nhân tố quyết định đối với sự phát triển của một công ty lập trình, phát triển game.

Trong số những thành công rực rỡ trong lĩnh vực phát triển trò chơi tại Việt Nam, không thể không kể đến cái tên **Nguyễn Hà Đông** – tác giả của trò chơi điện tử **Flappy Bird**. Mặc dù chỉ mới là một lập trình viên trẻ tuổi, anh đã làm được một việc khiến mọi nhà phát triển game đều mơ ước: "*Đó là sáng tạo ra một trò chơi khiến cả thế giới phải say mê*". Ứng dụng **Flappy Bird** chính là minh chứng cho sự thành công đó, trò chơi đã thu hút hơn 50 triệu lượt tải về và nhận được hơn 90,000 lượt xếp hạng trong gần 9 tháng hoạt động. Tuy nhiên, không lâu sau đó, **Flappy Bird** đã bị chỉ trích về mức độ khó của trò chơi, và đặc biệt là những cáo buộc về sao chép hình ảnh, âm thanh cũng như cách vận hành của một số trò chơi khác (*Super Mario, Piou Piou vs. Cactus,…)*. Cuối cùng, **Flappy Bird** đã bị chính cha đẻ của mình "khai tử" vào ngày 10/2/2014, đặt dấu chấm hết cho sự thành công của mình.

Với các đề tài phần mềm quản lý được giảng viên cung cấp trên lớp, chúng em muốn mang đến một đề tài mới mẻ, "độc" và "lạ". **Flappy Bird** chính là kết quả của sự quyết định đó. Nhóm chúng em muốn thử sức mình với việc lập trình phát triển game trên Windows Form để có thể có sự chuẩn bị tốt nhất trước khi bắt đầu chọn và học môn **Nhập môn Lập trình Game** trong các học phần tới. Sự đơn giản trong phần code tạo nên **Flappy Bird**, điều này sẽ hỗ trợ chúng em

trong những bước đầu quen với việc lập trình trò chơi. Nhưng với những tham vọng của từng thành viên trong nhóm, chúng em muốn mang lại một trò chơi **Flappy Bird** với một phiên bản hoàn toàn mới, thu hút hơn, "bắt mắt" hơn, và nhiều chức năng so với phiên bản gốc trước đó của nó.

Nhóm chúng em mong rằng với cơ hội được làm việc chung với nhau, chúng em tham vọng rằng sẽ không chỉ làm sống lại một **Flappy Bird** vốn đã từng là một biểu tượng thành công trong lĩnh vực phát triển game tại Việt Nam, mà còn giúp nó vươn xa hơn trên thị trường game quốc tế - như một sự tôn trọng dành cho cha đẻ của **Flappy Bird**.

MỤC LỤC

ĐỒ ÁN LẬP TRÌNH TRỰC QUAN – FLAPPY BIRD

CHƯƠNG I: GIỚI THIỆU

I – Đề tài

Chọn đề tài nằm ngoài danh sách giáo viên cung cấp, thống nhất chọn: *Flappy Bird.*

II – Lý do chọn đề tài

Nhận thấy tiềm năng của việc phát triển game trong nhu cầu thị trường lĩnh vực công nghệ thông tin cũng như khả năng kinh tế mà các trò chơi điện tử, ứng dụng giải trí mang lại. Ở giai đoạn hiện tại – sinh viên học kì thứ ba tại UIT, nhóm chúng em chỉ vừa mới làm quen ngôn ngữ C# và cách làm việc trên Windows Form.

Để có thể bắt đầu học cách lập trình phát triển game trước khi thực sự học môn *Nhập môn Lập trình Game*, **Flappy Bird** là một lựa chọn phù hợp và vừa sức với nhóm chúng em. Bởi vì code vận hành của game khá đơn giản với các chức năng dễ mô phỏng lại trên Windows Form, không đòi hỏi phần đồ họa cao, các phần mềm lập trình chuyên dụng và các tính năng phức tạp như các trò chơi hiện thành ngày nay.

**** Đối với cá nhân từng thành viên trong nhóm:***

Việc tìm hiểu sớm về các kiến thức lập trình phát triển game giúp từng người trong nhóm có thêm thông tin và sự chuẩn bị tốt nhất trước khi bắt đầu học môn *Nhập môn Lập trình Game*, bên cạnh đó giúp mỗi cá nhân đa dạng trong thao tác làm việc giữa các thuộc tính, sự kiện của Windows

Form và lập trình phát triển trò chơi điện tử để từ đó tạo ra một phiên bản **Flappy Bird** hoàn thiện.

Việc chọn một đề tài không có trong danh sách cung cấp sẵn là một thử thách và là động lực giúp chúng em chủ động tìm hiểu các kiến thức ngoài giáo trình, mở rộng vốn hiểu biết để từ đó linh hoạt trong việc vận dụng kiến thức vào thực tiễn.

**** Đối với xã hội:***

Với ý định ban đầu, nhóm chúng em chỉ mong có thể mang đến một sản phẩm đồ án hoàn thiện và chất lượng để có thể đạt kết quả tốt trong đợt báo cáo cuối kỳ môn *Lập trình trực quan* sắp tới.

Nhưng sau một thời gian làm việc, với sức trẻ, sự sáng tạo không ngừng của mỗi thành viên, chúng em ngày càng tham vọng nhiều điều hơn nữa. Với kết quả là một phiên bản **Flappy Bird** hoàn thiện, nhóm chúng em mong trò chơi này có thể một lần nữa trở thành những món ăn tinh thần cho người dùng, giúp họ giải trí sau những giờ làm việc áp lực và căng thẳng, nâng cao trải nghiệm người dùng và chất lượng cuộc sống.

Việc ngày càng hoàn thiện trò chơi và làm sống lại nó như là một cách thể hiện sự tôn trọng và ngưỡng mộ đối với cha đẻ của **Flappy Bird** – **Nguyễn Hà Đông**, người đã góp phần khẳng định khả năng của con người Việt Nam trong lĩnh vực công nghệ thông tin đến bạn bè quốc tế.

CHƯƠNG II: PHÂN TÍCH TÍNH KHẢ THI CỦA ĐỀ TÀI VÀ PHÂN CÔNG CÔNG VIỆC

I – Phân tích đề tài

- Đề tài:

Xây dựng trò chơi Flappy Bird.

- Phân tích:

+ *Tính phù hợp nội dung môn học:* Đã được giảng viên trên lớp duyệt qua đề tài.

+ *Tính khả thi:* Hoàn toàn khả thi do việc xây dựng **Flappy Bird** có thể xây dựng và thao tác trên Windows Form, dựa trên hướng dẫn và source code xây dựng trò chơi Flappy Bird trực tuyến.

Nguồn:https://www.youtube.com/watch?v=Zcvf7B0vvRI&fbclid=IwAR0juia3CYZUm_BwCPmzQ9Bch7i72eSXscmbH5vzQyaKJEOSkr1IcDXGb8o

II – Phân công công việc

- Nội dung công việc:

+ Tìm hiểu, phân tích cách xây dựng trò chơi trên Windows Form.

+ Phân tích, nắm được nguyên lý, cách thức vận hành của code xây dựng **Flappy Bird** theo hướng dẫn trực tuyến trên Youtbe.

+ Thực thi code theo phiên bản trực tuyến.

+ Phát triển trò chơi **Flappy Bird** thành một phiên bản mới, hoàn thiện, bổ sung các tính năng dựa trên việc lên ý tưởng, sáng tạo thêm các chức năng mới vốn chưa có trên phiên bản gốc.

+ Hiện thực hóa phiên bản mới.

+ Nhận xét và chỉnh sửa, hoàn thiện trò chơi.

- Phân công công việc:

+ Bảng phân công bao gồm các công việc chính và các ý tưởng sáng tạo được chia đều cho 4 thành viên theo khối lượng công việc và được trình bày chi tiết tại project đồ án của nhóm.

Nguồn:https://docs.google.com/spreadsheets/d/1nKklIpf0GDGW03y5Tlj-VlQasWvP3Msu2GTccWa6DVU/edit#gid=0

CHƯƠNG III: PHÂN TÍCH PHIÊN BẢN GỐC

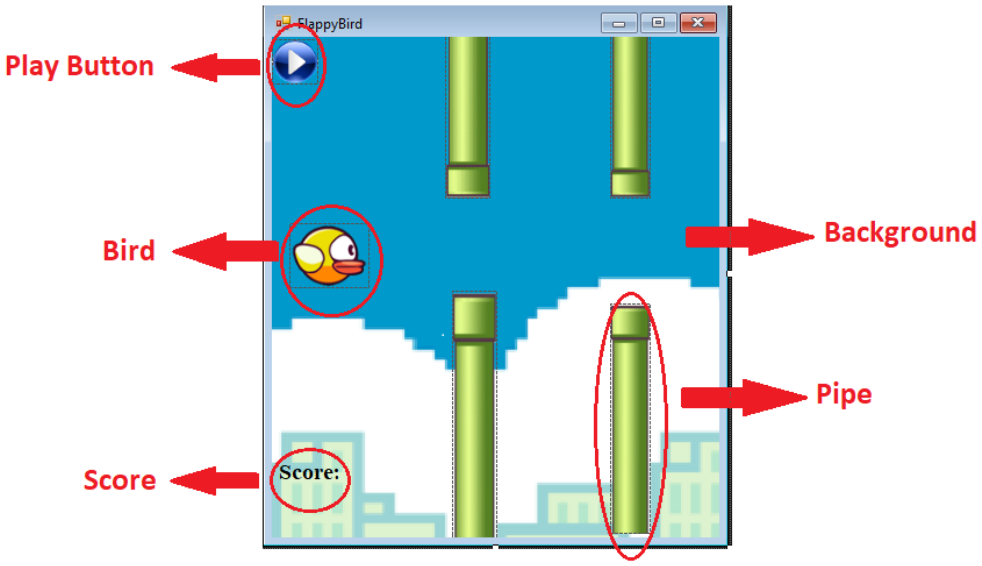
I – Phiên bản gốc

- Tài liệu hướng dẫn:

Nguồn:https://www.youtube.com/watch?v=Zcvf7B0vvRI&fbclid=IwAR0juia3CYZUm_BwCPmzQ9Bch7i72eSXscmbH5vzQyaKJEOSkr1IcDXGb8o

- Phân tích phiên bản gốc: liên qua các đối tượng và sự kiện cơ bản cần có.

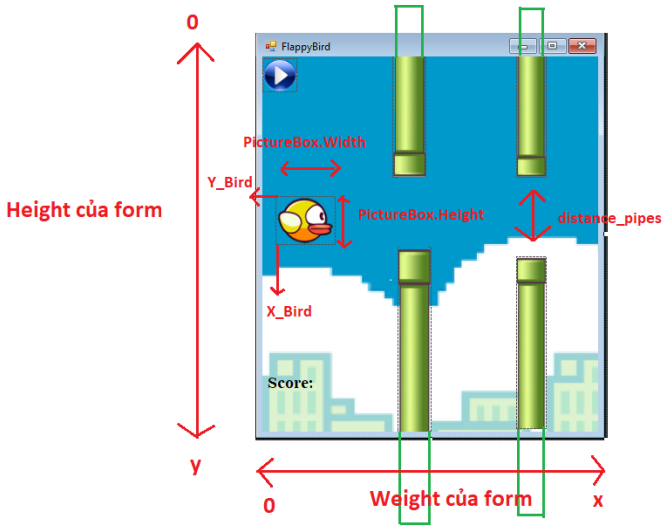
+ Các đối tượng cơ bản:



- ***Bird***: đối tượng chính trong trò chơi, di chuyển qua khe giữa đối tượng *Pipe* để tăng điểm và tiếp tục trò chơi. Đối tượng *Bird* xây dựng bằng *PictureBox* thông qua việc thêm ảnh tĩnh.
- ***Pipe****:* đối tượng vật cản trong trò chơi, gây cản trở việc di chuyển của đối tượng *Bird*. Nếu đối tượng *Bird* va chạm với đối tượng *Pipe*, trò chơi sẽ kết thúc. Đối tượng *Pipe* xây dựng bằng *PictureBox* thông qua việc thêm ảnh tĩnh.

- ***Score***: đối tượng ghi nhận điểm số và tăng điểm số tương ứng mỗi khi đối tượng *Bird* đi qua thành công khe của đối tượng *Pipe*. Đối tượng *Score* xây dựng bằng *Label* thông qua code thay đổi text của *Label*.
- ***Play Button***: xây dựng bằng *Button*, có chức năng bắt đầu trò chơi sau khi người dùng click vào.
- ***Background***: đối tượng làm nền của trò chơi. *Background xây dựng bằng thuộc tính có sẵn của Form là BackgroundImage* thông qua việc thêm ảnh tĩnh.

+ *Các sự kiện cơ bản:* Tọa độ *X* của *Form* được bắt đầu tính từ bên trái *Form* theo hướng trục hoành sang phải., tọa độ *Y* được bắt đầu tính từ bên trên cùng của *Form* theo hướng trục tung đi xuống. Dùng phím *Space* để cho đối tượng *Bird* bay lên (giảm tung độ *Y_Bird*).



- ***Draw Default Pipe**:* có 4 đối tượng *Pipe* tham gia vào việc xây dựng trò chơi lần lượt là *PipeAbove1, PipeBottom1, PipeAbove2, PipeBottom2*. Các cặp *Pipe* đối xứng trên cùng có thể được gom thành một *PipePair* để xét hoàng độ do có cùng hoành độ với nhau, ta có hai *PipePair* ứng với 4 *Pipe*. Tọa độ cặp *PipePair1* được xác định ứng với hoành độ chung là *X_PipePair1* và lần lượt ứng với *PipeAbove1* và *PipeBottom1* ta có được tung độ tưng ứng. Tương tự khi xét tọa độ cho *PipePair2*. Giữa một *PipePair* bất kì luôn tồn tại một khoảng cách cố định là *distance_pipes*, và giữa hai cặp *PipePair* bất kì cũng luôn cách nhau một khoảng cố định. Trong quá trình chơi, chỉ có *X_PipePair* là luôn giảm theo *Timer* và luôn được vẽ lại ứng với tọa độ mới tạo nên sự chuyển động trên màn hình.
- ***Draw Random Pipe**:* để thuận lợi cho việc xét *Random Pipe* trên màn hình, *PictureBox* của *Pipe* luôn được xét nằm ngoài tung độ của *Form*, nghĩa là *Y_Pipe* luôn bé hơn 0. Với trục hoành, sau một khoảng thời gian chơi nếu X_PipePair bất kì bé hơn 0, sẽ cộng một khoảng hoành độ phù hợp để thiết lập tọa độ mới cho *PipePair* đó sao cho nằm ngoài *Form* về phía bên phải, và theo *Timer* thì *X_PipePair* luôn giảm và tạo nên một sự chuyển động mượt. Kết quả là *PipePair cũ* xuất hiện lại trên màn hình nhưng được xem như là một *PipePair* mới. Với trục tung, ứng với mỗi khi đẩy một *PipePair* sang bên phải, sẽ xét *Random* kích thước *Pipe* thông qua việc cộng trừ một khoảnh tung độ ngắn cho *Y_Pipe* của đối tượng *Pipe* tương ứng, điều này tạo nên những đối tương *Pipe* có kích thước dài ngắn khác nhau.

- ***Va chạm giữa Bird và Pipe**:* Với một *PipePair* bất kì, ta luôn có hai trường hợp là đối tượng *Bird* va chạm đối tượng *PipeAbove* hoặc va chạm đối tượng *PipeBottom*. Điều kiện xác định va chạm được kết luận như sau:

  Vào điều kiện có khả năng xảy ra va chạm:

  (X_Bird + PictureBox(Bird).Width >= X_PipePair) && (X_Bird + PictureBox(Bird).Width <= X_PipePair + PictureBox(Pipe).Width) (1)

  Nếu đi vào điều kiện trên, khả năng va chạm ống trên xảy ra khi và chỉ khi:

  (Y_Bird <= Y_PipeAbove + PictureBox(Pipe).Height) (2)

Ngược lại, khả năng va chạm ống dưới xảy ra khi và chỉ khi:
(Y_Bird + PictureBox(Bird).Height >= Y_PipeBottom) (3)

Nếu ***(1) (2)*** hoặc ***(1) (3)*** xảy ra, *Timer* của *Form* sẽ dừng lại, đồng nghĩa người chơi đã *Game Over*.

CHƯƠNG IV: Ý TƯỞNG VÀ HIỆN THỰC HÓA THÀNH PHIÊN BẢN MỚI

I – Ý tưởng

- Một giao diện mới hấp dẫn hơn, dễ sử dụng và thao tác cho người dùng.

- Tạo hiệu ứng động cho một số đối tượng: *Bird, …*

- Ứng với thao tác chọn button, tăng điểm, vv… sẽ có âm thanh tương ứng.

- Thêm một số tính năng mới:

+ Thêm nhiều button đa dạng hóa trong việc cài đặt, chỉnh sửa đồ họa hoặc tăng giảm độ khó khi chơi theo ý muốn người dùng.

+ Thêm một số item để nâng cao trải nghiệm người dùng: *Hearts, Coins, Rockets, Gifts, vv…* Ứng với mỗi item đều có các chức năng tương tác với đối tượng *Bird*

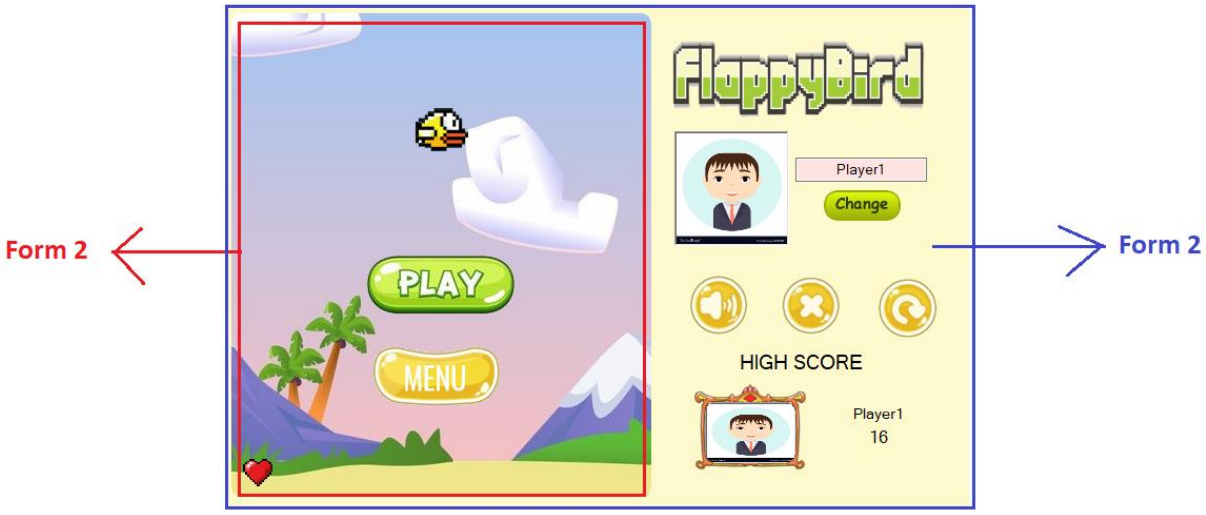
+ Tự động ghi nhận và cập nhật high score trên màn hình ứng với từng tên người chơi.

…

II – Hiện thực hóa

1. Giao diện mới

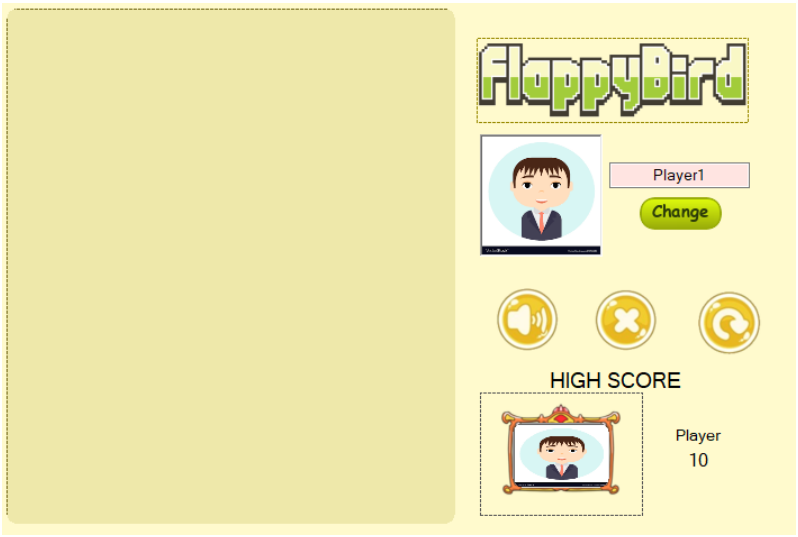
- Giao diện mới được xây dựng dựa trên hai *Form* riêng biệt, hai *Form* được sắp xếp một vị trí hợp lý để tạo nên một giao diện chung duy nhất ngay giữa màn hình người dùng.



1.1 Form 1

- Là màn hình giao diện dành cho việc trình bày thông tin game, như là tên game, các *Button* tăng giảm âm lượng, thoát trò chơi và *replay*, ngoài ra còn có tên người chơi, hình ảnh người chơi cung cấp, thông tin được lưu lại về người chơi có số điểm cao nhất bao gồm hình ảnh, tên người chơi và số điểm đạt được.

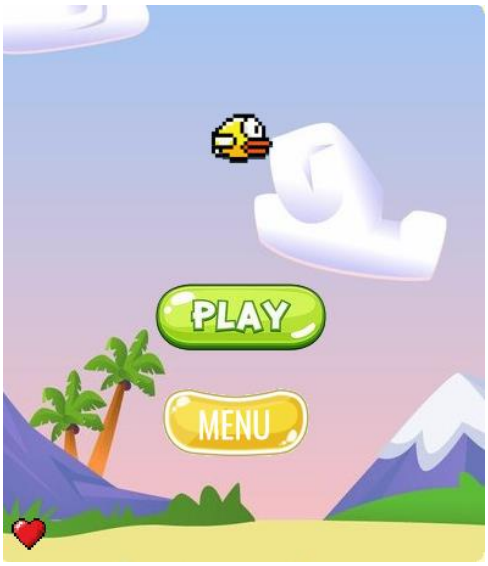
- *Replay Button* có chứ năng chạy lại *Form* để tiến hành việc chơi lại.



1.2 Form 2:

- Màn hình chính của game, bao gồm các *Button: Play, Menu*. Ngoài ra còn thể hiện *Background* chính của game và hình ảnh động của *Flappy Bird* tạo sự thu hút cho người dùng.

- *Form 2* là màn hình hiển thị cụ thể quá trình chơi và hàng loạt các sự kiện khác có liên quan.



2. Thêm button

2.1 Các Button trong Menu:

2.1.1 Season:

- Sau khi click vào *Button* này, các *Button* trên màn hình sẽ ẩn đi. Trên *Panel* xuất hiện ba *Button* mới lần lượt là các chủ đề *Original, Christmas, Halloween* để lựa chọn, tùy vào mỗi lựa chọn của người dùng mà chương trình thay đổi hình ảnh các đối tượng trong trò chơi theo chủ đề tương ứng (bao gồm *Background, Pipe, Gift, Rocket,…)*. Sau khi đã chọn xong, màn hình sẽ quay trở lại giao diện trước đó.

- Sẽ có âm thanh *Click Button* mỗi khi thao tác.

Ví dụ khi chọn chủ đề Christmas

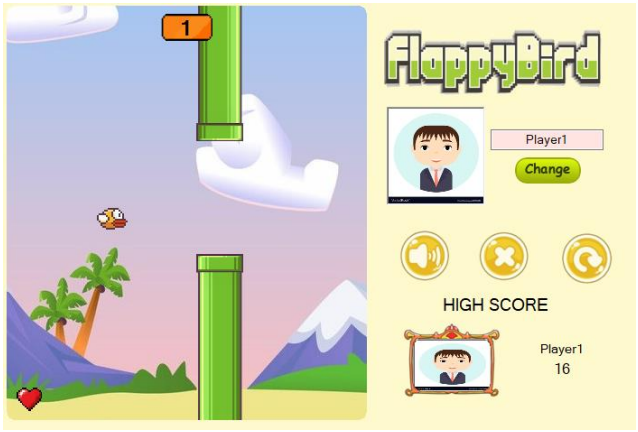
2.1.2 Item:

- Sau khi click vào *Button* này, các *Button* trên màn hình sẽ ẩn đi. Trên *Panel* hiện tại là ba *PictureBox* nhân vật đang chuyển động để lựa chọn, tùy vào mỗi lựa chọn của người dùng mà chương trình sẽ hiển thị nhân vật trong game tương ứng. Trong quá trình chơi game, các nhân vật đều có hiệu ứng *animation*. Sau khi chọn xong nhân vật, màn hình sẽ quay trở lại giao diện chính ban đầu.

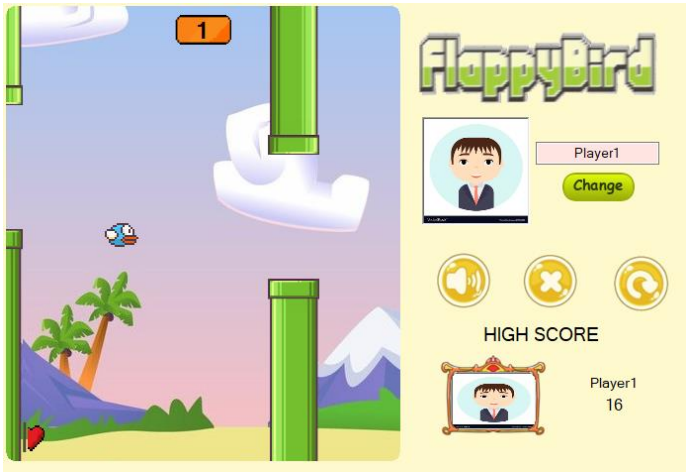
- Sẽ có âm thanh *Click Button* mỗi khi thao tác.

- Sẽ có một biến *index_bird* kiểu số nguyên nắm vai trò giữ vị trí cho việc lựa chọn nhân vật. Giá trị khởi tạo mặc định là *1* ứng với việc người dùng vào chơi ngay mà không lựa chọn nhân vật, nhân vật mặc định khi chơi sẽ là *Yellow Bird*. Trong trường hợp lựa chọn nhân vật, lần lượt các index_bird sẽ cập nhật giá trị như sau:

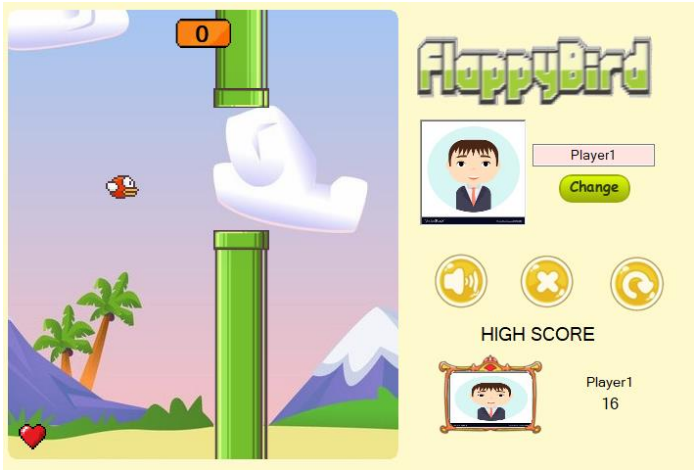
+ *PictureBox1*: cập nhật giá trị *index_bird* = 1 (ứng với *Yellow Bird* trong game – nhân vật mặc định).



+ *PictureBox2*: cập nhật giá trị *index_bird* = 2 (ứng với *Blue Bird* trong game).



+ *PictureBox3*: cập nhật giá trị *index_bird* = 3 (ứng với *Red Bird* trong game).

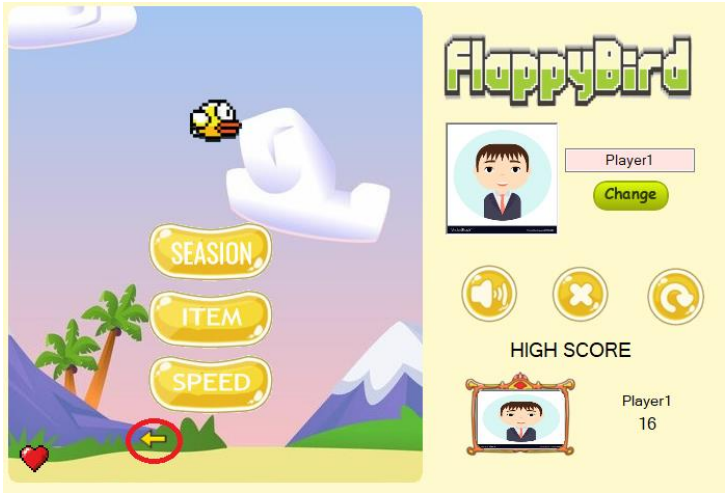


2.1.3 Speed:

- Sau khi click vào *Button* này, các *Button* trên màn hình sẽ ẩn đi. Trên *Panel* là ba *Button* mới lần lượt là *Easy*, *Medium*, *Hard* để lựa chọn, tùy vào mỗi lựa chọn của người dùng mà chương trình thay đổi độ khó trò chơi thông qua việc tăng giảm tốc độ di chuyển của các đối tượng trên màn hình. Sau khi đã chọn xong, màn hình sẽ quay trở lại giao diện trước đó.

- Sẽ có âm thanh *Click Button* mỗi khi thao tác.

2.1.4 Button Narrow: *Button* có hình dạng mũi tên hướng sang bên trái, sau khi click *Button này*, chương trình sẽ cho quay lại giao diện màn hình trước đó. Sẽ có âm thanh *Click Button* mỗi khi thao tác.



2.2 Button Play: Sau khi click *Button* này, trò chơi sẽ được bắt đầu. Sẽ có âm thanh *Click Button* mỗi khi thao tác.

3. Thêm item:

3.1 Coin:

3.1.1 Chức năng: cộng 3 điểm vào phần *Score*.

3.1.2 Thực thi chương trình:

3.1.2.1 Biến khởi tạo:

```
int X_Coins;
int Y_Coins;
int coins_Count = 0;
int distance_2pipe = 150;
int randomNext_Coins = 3;
```

3.1.2.2 Draw Coin:

- *Coin* được cho xuất hiện trong khoảng ngẫu nhiên từ 4 đến 5, ứng với việc đã đi qua 4 hoặc 5 *Pipe* tính từ lần *Coin* xuất hiện trước đó. Biến *coins_Count* sẽ làm tín hiệu, ứng với mỗi lần *Bird* đi qua được một *Pipe* thì biến *coins_Count* sẽ tăng lên 1 và sẽ reset lại bằng *0* nếu thỏa điều kiện xuất hiện.

- Sau khi thỏa điều kiện trên, *Coin* sẽ được cho phép xuất hiện (*Visible = true)* thông qua việc cập nhật tọa độ mới lần lượt ứng

với *X_Coins* và *Y_Coins* sao cho *Coin* xuất hiện ngẫu nhiên và hợp lý, không đè lên hình *Pipe* hoặc xuất hiện ngoài màn *Form 2*.

3.1.2.3 Coin in game:

- *Coin* sẽ giảm dần *X_Coins* ứng với *Timer*, và luôn cập nhật vị trí mới để tạo nên hiệu ứng *animation* trong quá trình chơi.

3.1.2.4 Eat Coin:

- Đối tượng *Bird* được xác đinh là đã ăn *Coin* khi và chỉ khi giữa *PictureBox* của cả hai tiếp xúc, va chạm với nhau.

- Điều kiện xác định nằm trong khoảng có khả năng va chạm:

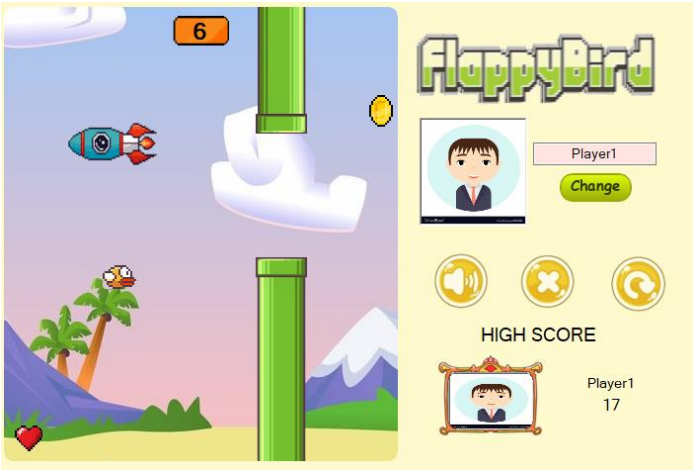
(X_Bird + PictureBox(Bird).Width >= X_Coins) &&
(X_Bird <= X_Coins + PictureBox(Coin).Height (1)

- Nếu điều kiện trên là đúng, trường hợp xảy ra va chạm khi và chỉ khi:

(Y_Bird + PictureBox(Bird).Height >= Y_Coins)
&& (Y_Bird <= Y_Coins + PictureBox(Coin).Height) (2)

- Nếu **(1), (2)** xảy ra, đối tượng *Bird* được xác định là đã ăn *Coin*. Sau khi ăn, *Coin* sẽ biến mất thông qua việc cho *Visible = false* và sẽ xuất hiện trở lại màn hình (*Visible = true)* cho đến khi sự kiện *Draw Coin* diễn ra.

- Khi ăn được *Coin*, *Score* của người chơi sẽ tăng lên 3 điểm, đồng thời sẽ có hiệu ứng âm thanh riêng cho việc đã ăn được *Coin* trong trò chơi.



3.2 Heart:

3.2.1 Chức năng: thêm một mạng chơi cho đối tượng *Bird*. Đối tượng *Bird* được khởi tạo với giá trị mạng ban đầu là *1*, ứng với mỗi mạng sau khi được tăng bởi *Heart*, chương trình sẽ tạo một đối tượng *Shield* có khả năng che chắn cho đối tượng *Bird* trước cản vật cản. Nếu đối tượng *Shield* còn tồn tại (số mạng lớn hơn *1*), thì trò chơi vẫn được tiếp tục cho dù có xảy ra các sự kiện va chạm dẫn đến *Game Over*. Nếu đối tượng *Shield* chắn thành công, mạng sẽ trừ đi một, và vật cản sẽ biến mất. Nếu số mạng của đối tượng *Bird* bằng *1*, đối tượng *Shield* sẽ biến mất, các sự kiện va chạm *Game Over* sẽ hoạt động trở lại.

3.2.2 Thực thi chương trình:

3.2.2.1 Biến khởi tạo:

```
///Hearts
int X_Hearts;
int Y_Hearts;
int pipes_Count = 0;
int distance_2pipe = 150;
int randomNext_Hearts = 3;
```

```
public bool sign_getHearts = false;
public bool sign_decreaseHearts = false;

///Shields
int X_Shields;
int Y_Shields;
```

3.2.2.2 Draw Heart:

- *Heart* được cho xuất hiện trong khoảng ngẫu nhiên từ 8 đến 9, ứng với việc đã đi qua 8 hoặc 9 *Pipe* tính từ lần *Heart* xuất hiện trước đó. Biến *pipes_Count* sẽ làm tín hiệu, ứng với mỗi lần *Bird* đi qua được một *Pipe* thì biến *pipes_Count* sẽ tăng lên 1 và sẽ reset lại bằng *0* nếu thỏa điều kiện xuất hiện.

- Sau khi thỏa điều kiện trên, *Heart* sẽ được cho phép xuất hiện (*Visible = true)* thông qua việc cập nhật tọa độ mới lần lượt ứng với *X_Hearts* và *Y_Hearts* sao cho *Heart* xuất hiện ngẫu nhiên và hợp lý, không đè lên hình *Pipe* hoặc xuất hiện ngoài màn *Form 2*.

3.2.2.3 Heart in game:

- *Heart* sẽ giảm dần *X_Hearts* ứng với *Timer,* và luôn cập nhật vị trí mới để tạo nên hiệu ứng *animation* trong quá trình chơi.

3.2.2.4 Eat Heart:

- Đối tượng *Bird* được xác đinh là đã ăn *Heart* khi và chỉ khi giữa *PictureBox* của cả hai tiếp xúc, va chạm với nhau.

- Điều kiện xác định nằm trong khoảng có khả năng va chạm:

(X_Bird + PictureBox(Bird).Width >= X_Hearts) &&
(X_Bird <= X_Hearts + PictureBox(Heart).Height (1)

- Nếu điều kiện trên là đúng, trường hợp xảy ra va chạm khi và chỉ khi:

(Y_Bird + PictureBox(Bird).Height >= Y_Hearts)
&& (Y_Bird <= Y_Hearts + PictureBox(Heart).Height) (2)

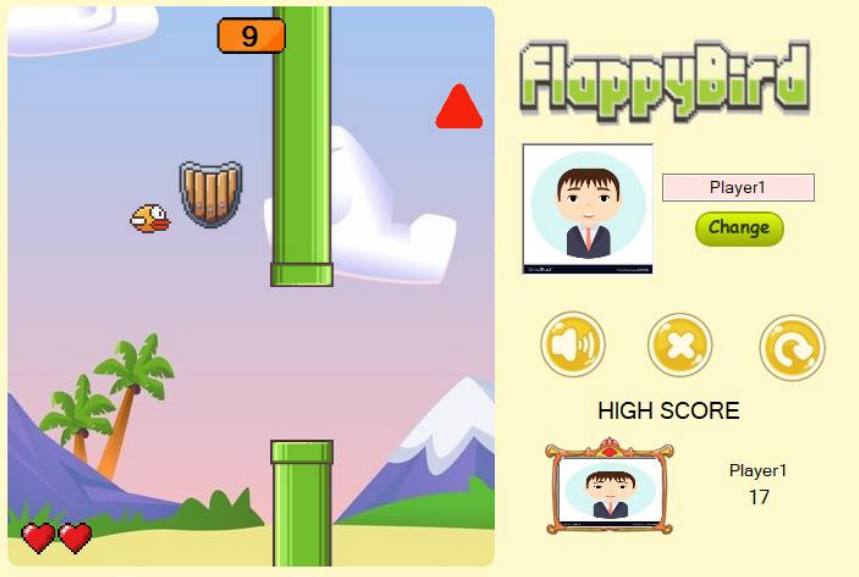
- Nếu **(1), (2)** xảy ra, đối tượng *Bird* được xác định là đã ăn *Heart*. Sau khi ăn, *Heart* sẽ biến mất thông qua việc cho *Visible = false* và sẽ xuất hiện trở lại màn hình (*Visible = true)* cho đến khi sự kiện *Draw Heart* diễn ra.

- Khi ăn được *Heart*, chương trình sẽ truy cập vào *Class LifeSpan* để tăng cho biến *iHearts* bên trong lên *1*, giá trị *iHearts* luôn được cố định trong khoảng từ 1 đến 3. Ứng với việc giá trị *iHearts* lớn hơn *1*, trò chơi sẽ tạo ra một đối tượng *Shield* trước *Bird*.

- Ngoài ra, khi trò chơi bắt đầu, sẽ có một *PictureBox* của *Heart* được vẽ ở góc phía dưới bên trái màn hình để cho người chơi biết giá trị mặc định của số mạng đối tượng *Bird* khi bắt đầu là *1*. Ứng với mỗi lần *Eat Heart*, sẽ xuất hiện thêm một *PictureBox* của *Heart* mới bên cạnh, tối đa là ba *PictureBox* của *Heart*.

- Số lượng *PictureBox* của *Heart* sẽ giảm nếu đối tượng *Shield* chắn vật cản thành công.

- Sẽ có âm thanh tương ứng với sự kiện *Eat Heart* thành công.



Số mạng Bird = 2 (iHearts = 2)

Số mạng Bird = 3 (iHearts = 3)

3.2.2.5 Draw Shield:

- *Shield* được cho xuất hiện với điều kiện cố định là biến *iHearts* trong *Class LifeSpan* lớn hơn 1, ứng với đã diễn ra sự kiện *Eat Heart*. Biến *sign_getHearts* sẽ làm tín hiệu, ứng với mỗi lần sự kiện *Eat Heart* xảy ra, *sign_getHearts* sẽ trả về giá trị *true*, và sẽ trả về giá trị *false* sau khi đã thực hiện xong việc tăng giá trị cho biến *iHearts* trong *Class LifeSpan*.

- Sau khi thỏa điều kiện trên, *Shield* sẽ được cho phép xuất hiện (*Visible = true)* thông qua việc cập nhật tọa độ mới lần lượt ứng với

X_Shields và *Y_Shields* sao cho luôn cố định trước đối tượng *Bird*, cụ thể là gán giá trị cho hai biến lần lượt là:

X_Shield = X_Bird + 50;

Y_Shield = Y_Bird - 30;

3.2.2.6 Shield in game:

- Ứng với vị trí của đối tượng *Bird*, *Shield* luôn cập nhật một vị trí cố định phía trước *Bird* sao cho thỏa điều kiện biến *iHearts* trong *Class LifeSpan* lớn hơn *1*.

3.2.2.7 Shield impacts *Other Objects*:

- Điều kiện tiên quyết để xảy ra sự kiện này là đối tượng *Shields* phải xuất hiện trên màn hình.

- Đối tượng *Shield* được xác đinh là chắn cho *Bird* thành công khi giữa *PictureBox* của *Shield* và của các *Other Objects (Pipes, Rockets)* tiếp xúc, va chạm với nhau.

- Điều kiện xác định nằm trong khoảng có khả năng va chạm:

***Đối với va chạm Rockets:**

(X_Shield + PictureBox(Shield).Width >= X_Rocket) && (X_Shield <= X_Rocket + PictureBox(Rocket).Height (1)

*** Đối với va chạm Pipes:**

(X_Shield + PictureBox(Shield).Width >= X_PipePair) && (X_Shield + PictureBox(Shield).Width <= X_PipePair + PictureBox(Pipe).Width) (2)

- Nếu điều kiện trên là đúng, trường hợp xảy ra va chạm khi và chỉ khi:

***Đối với va chạm Rockets:**

(Y_Shield + PictureBox(Shield).Height >= Y_Rocket) && (Y_Shield <= Y_Rocket + PictureBox(Rocket).Height) (3)

*** Đối với va chạm Pipes:**

+ Nếu đi vào điều kiện trên, khả năng va chạm ống trên xảy ra khi và chỉ khi:

(Y_Shield <= Y_PipeAbove + PictureBox(Pipe).Height) (4)

+ Ngược lại, khả năng va chạm ống dưới xảy ra khi và chỉ khi:

(Y_Shield + PictureBox(Shield).Height >= Y_PipeBottom) (5)

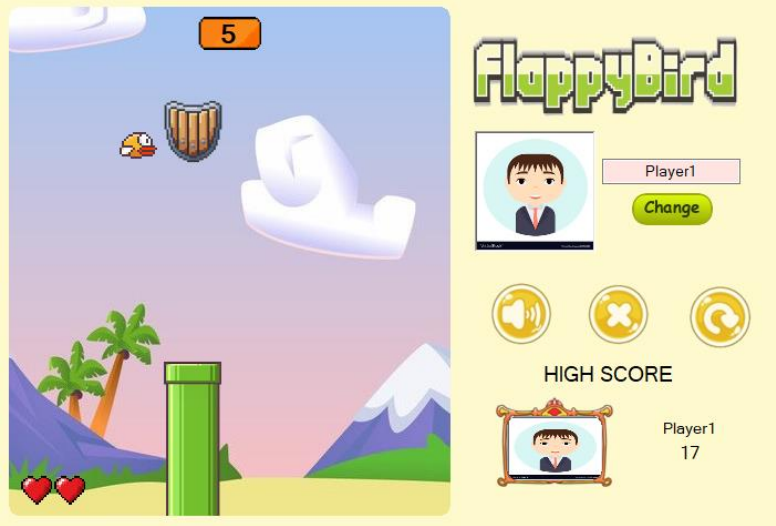
- Nếu **(1), (3)** xảy ra, đối tượng *Shield* được xác định là đã chắn đối tượng *Rocket, trong trường hợp* **(2), (4)** hoặc **(2), (5)** thì được xác định là đã chắn đối tượng *Pipe.* Sau khi chắn thành công, *Shield* sẽ biến mất nếu giá trị biến *iHearts* trong *Class LifeSpan* là 1. Ứng với mỗi lần chắn thành công, giá trị biến *iHearts* trong *Class LifeSpan* giảm đi *1*, đối tượng vật cản khi bị *Shield* chắn sẽ biến mất, đồng nghĩa việc gán cho *Other Objects* giá trị *Visible = false.*

- Khi *Shield* va chạm với các đối tượng vật cản, đối tượng vật cản sẽ biến mất, đối tượng *Bird* có thể đi xuyên qua một cách bình thường.

- Khi *Shield* chắn thành công, sẽ có một *PictureBox* của *Heart* bị biến mất đi ở góc bên trái phía dưới màn hình.

- Sẽ có âm thanh tương ứng với sự kiện *Shield impacts Other Objects* thành công.

- Ngoài ra, trong trường hợp đối tượng *Shield* vẫn còn tồn tại trên màn hình nhưng đối tượng *Bird* lại xảy ra va chạm (*PictureBox* của *Bird* tiếp xúc với các *PictureBox* vận cản), thì trò chơi vẫn được tiếp tục bằng cách thực hiện kết quả của sự kiện *Shield impacts Other Objects.*



Va chạm giữa Shield và Pipe

3.3 Rocket:

3.3.1 Chức năng: đối tượng *Rocket* được xem là một trong các chướng ngại vật trong trò chơi, nếu đối tượng *Rocket* va chạm trực tiếp với đối tượng *Bird*, trong trường hợp số mạng hiện tại của đối tượng *Bird* bằng *1*, người chơi sẽ *Game Over* (trò chơi dừng lại).

3.3.2 Thực thi chương trình:

3.3.2.1 Biến khởi tạo:

```
public int X_Rocket;
public int Y_Rocket;
int rocket_Count = 0;
int randomNext_Rocket = 4;
int distance_2pipe = 150;

int X_Fire;
int Y_Fire;
```

3.3.2.2 Draw Emergency Symbol:

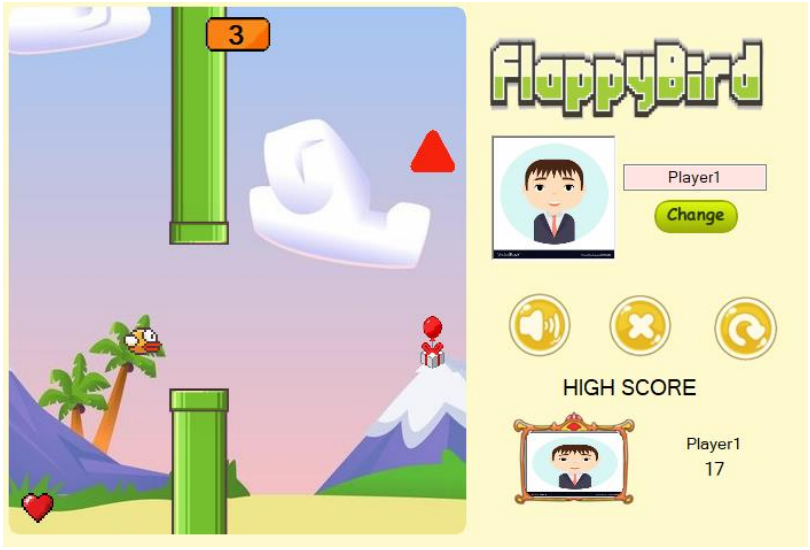
- *Emergency Symbol* được cho xuất hiện trong khoảng ngẫu nhiên từ 9 đến 10, ứng với việc đã đi qua 9 hoặc 10 *Pipe* tính từ lần

Emergency Symbol xuất hiện trước đó. Biến *rocket_Count* sẽ làm tín hiệu, ứng với mỗi lần *Bird* đi qua được một *Pipe* thì biến *rocket_Count* sẽ tăng lên 1 và sẽ reset lại bằng *0* nếu thỏa điều kiện xuất hiện.

- Sau khi thỏa điều kiện trên, *Emergency Symbol* sẽ được cho phép xuất hiện như là một PictureBox với *Visible = true* tại một tọa độ có hoành độ *cố định* và chỉ dịch chuyển tung độ lên xuống ngẫu nhiên *để gây khó khăn cho người chơi, Emergency Symbol* xuất hiện sát bên phải màn hình *Form 2*.

- *Emergency Symbol* sẽ biến mất khi đối tượng *Rocket* xuất hiện, nghĩa là gán *Visible = false* và sẽ có giá trị *true* trở lại nếu thỏa điều kiện trên.

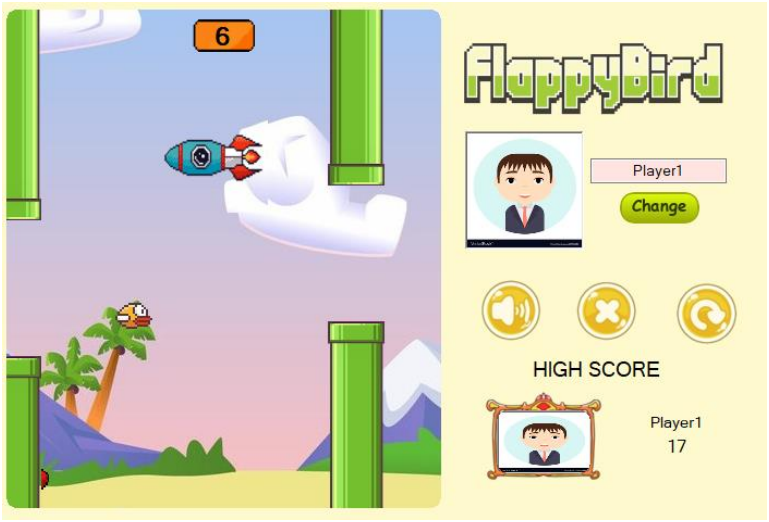
- Có âm thanh và các hiệu ứng động tương ứng.



3.3.2.3 Draw Rocket:

- *Rocket* được cho xuất hiện sau khi sự kiện *Draw Emergency Symbol* xảy ra, cụ thể là khi *PictureBox* của *Emergency Symbol* gán *Visible = false*.

- Sau khi thỏa điều kiện trên, *Rocket* sẽ được cho phép xuất hiện (*Visible = true)* thông qua việc cập nhật tọa độ mới lần lượt ứng với *X_Rocket* và *Y_Rocket* tại vị trí *Emergency Symbol* xuất hiện.



3.3.2.4 Rocket in game:

- *Rocket* sẽ giảm dần *X_Rocket* ứng với *Timer*, và luôn cập nhật vị trí mới để tạo nên hiệu ứng *animation* trong quá trình chơi.

3.3.2.5 Rocket impacts Bird:

- Đối tượng *Bird* được xác đinh là đã va chạm với đối tượng *Rocket* khi và chỉ khi giữa *PictureBox* của cả hai tiếp xúc, va chạm với nhau.

- Điều kiện xác định nằm trong khoảng có khả năng va chạm:

(X_Bird + PictureBox(Bird).Width >= X_Rocket) &&
(X_Bird <= X_Rocket + PictureBox(Rocket).Height (1)

- Nếu điều kiện trên là đúng, trường hợp xảy ra va chạm khi và chỉ khi:

(Y_Bird + PictureBox(Bird).Height >= Y_Rocket)
&& (Y_Bird <= Y_Rocket + PictureBox(Rocket).Height)
(2)

- Nếu **(1), (2)** xảy ra, đối tượng *Bird* được xác định là đã va chạm *Rocket.* Sau khi va chạm, *Rocket* sẽ biến mất thông qua việc cho *Visible = false* và sẽ xuất hiện trở lại màn hình (*Visible = true)* cho đến khi sự kiện *Draw Rocket* diễn ra.

- Khi va chạm *Rocket*, người chơi sẽ *Game Over*, đồng nghĩa chương trình sẽ dừng lại. Trước khi dừng lại, sẽ có hiệu ứng va chạm giữa đối tượng *Bird* và đối tượng *Rocket*. *Bird* khi va chạm sẽ quay một góc *-90* độ (quay xuống) và sẽ có hiệu ứng animation *Fire* để miêu tả đối tượng *Rocket* đã làm bốc cháy đối tượng *Bird* thông qua cập nhật tọa độ *PictureBox* của *Fire* với *X_Fire*, *Y_Fire* lần lượt ứng với *X_Bird*, *Y_Bird* theo *Timer*.

- Ngoài ra, trong trường hợp đối tượng *Shield* đang tồn tại trên màn hình, chương trình sẽ thi hành sự kiện *Shield impacts Other Objects* trước khi xét các sự kiện dẫn tới *Game Over*.

- Số lượng *PictureBox* của *Heart* sẽ giảm nếu đối tượng *Rocket* bị đối tượng *Shield* chắn thành công.

- Sẽ có âm thanh tương ứng với sự kiện va chạm (giữa *Bird* và *Rocket*, giữa *Shield* và *Rocket*, *Fire*).

3.4 Gift:

3.4.1 Chức năng: là *item* hỗ trợ cho đối tượng *Bird*, giúp đối tượng *Bird* tăng tốc độ bay và và có khả năng đi xuyên qua các đối tượng *Pipe*, nhưng *item* này không có tác dụng với đối tượng *Rocket*.

3.4.2 Thực thi chương trình:

3.4.2.1 Biến khởi tạo:

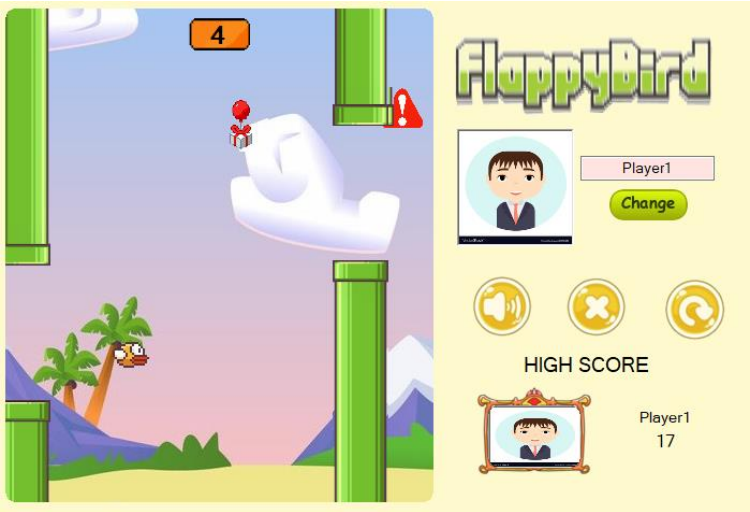
```
int X_Gift;
int Y_Gift;
int gift_Count = 0;
```

```
int randomNext_Gift = 4;
int distance_2pipe = 150;
public int count_Flash = 0;
```

3.4.2.2 Draw Gift:

- *Gift* được cho xuất hiện trong khoảng ngẫu nhiên từ 12 đến 13, ứng với việc đã đi qua 12 hoặc 13 *Pipe* tính từ lần *Gift* xuất hiện trước đó. Biến *gift_Count* sẽ làm tín hiệu, ứng với mỗi lần *Bird* đi qua được một *Pipe* thì biến *gift_Count* sẽ tăng lên 1 và sẽ reset lại bằng *0* nếu thỏa điều kiện xuất hiện.

- Sau khi thỏa điều kiện trên, *Gift* sẽ được cho phép xuất hiện (*Visible = true)* thông qua việc cập nhật tọa độ mới lần lượt ứng với *X_Gift* và *Y_Gift* sao cho *Gift* xuất hiện ngẫu nhiên và hợp lý, không đè lên hình *Pipe* hoặc xuất hiện ngoài màn *Form 2*.



3.4.2.3 Gift in game:

- *Gift* sẽ giảm dần *X_Gift* ứng với *Timer*, và luôn cập nhật vị trí mới để tạo nên hiệu ứng *animation* trong quá trình chơi.

3.4.2.4 Eat Gift:

- Đối tượng *Bird* được xác đinh là đã ăn *Gift* khi và chỉ khi giữa *PictureBox* của cả hai tiếp xúc, va chạm với nhau.

- Điều kiện xác định nằm trong khoảng có khả năng va chạm:

(X_Bird + PictureBox(Bird).Width >= X_Gift) &&
(X_Bird <= X_Gift + PictureBox(Gift).Height (1)

- Nếu điều kiện trên là đúng, trường hợp xảy ra va chạm khi và chỉ khi:

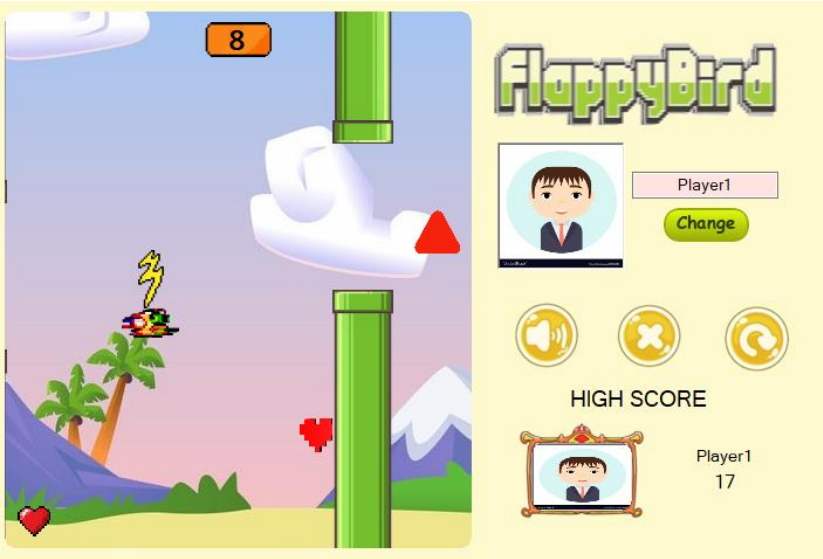
(Y_Bird + PictureBox(Bird).Height >= Y_Gift)
&& (Y_Bird <= Y_Gift + PictureBox(Gift).Height) (2)

- Nếu **(1), (2)** xảy ra, đối tượng *Bird* được xác định là đã ăn *Gift*. Sau khi ăn, *Gift* sẽ biến mất thông qua việc cho *Visible = false* và sẽ xuất hiện trở lại màn hình (*Visible = true)* cho đến khi sự kiện *Draw Gift* diễn ra.

- Khi ăn được *Gift*, sẽ có một biến *isGetGift* trong *Class Bird* làm tín hiệu và gán giá trị bằng *true*. Trong khi giá trị biến *isGetGift* này vẫn bằng *true*, mọi sự kiện va chạm *Pipe* dẫn tới *Game Over* đều bị bỏ qua, chỉ xét các điều kiện bình thường khi va chạm với đối tượng *Rocket*. Biến *isGetGift* sẽ trả về giá trị *false* sau một khoảng thời gian cố định. Kết quả là đối tượng *Bird* sẽ có khả năng đi xuyên qua ống trong khoảng thời gian đó.

- Ngoài ra, khi ăn được *Gift*, đối tượng *Bird* sẽ được thay đổi hình ảnh nhân vật, tăng tốc độ bay và xuất hiện thêm *Thunder Symbol* như là dấu hiệu cho biết hiệu ứng *Gift Item đang diễn ra.*

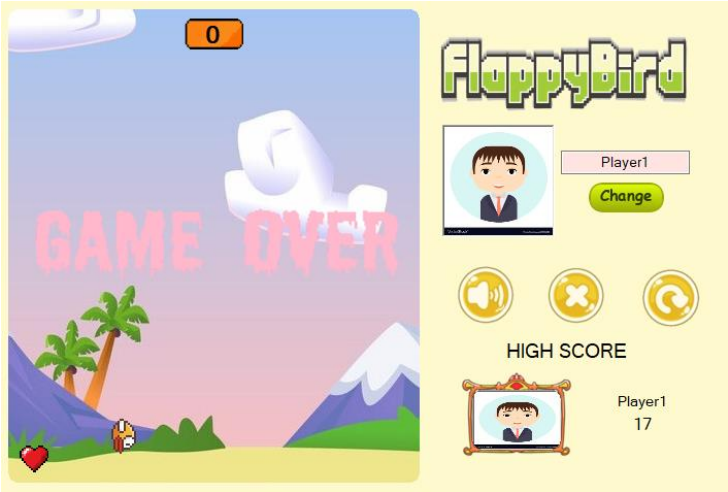
- Sẽ có âm thanh tương ứng với hiệu ứng của *Gift Item* đang diễn ra.



4. Game Over:

- Sự kiện *Game Over* diễn ra khi giá trị *iHearts* trong *Class LifeSpan* của đối tượng *Bird* bằng 1, đồng thời thỏa các điều kiện xảy ra va chạm với các đối tượng khác (*Bird* va chạm với các đối tượng vật cản).

- Khi sự kiện này diễn ra, *Timer* của chương trình sẽ ngừng lại (dừng chương trình), đối tượng *Bird* sẽ được cho quay một góc *-90* độ (quay hướng xuống) tạo hình ảnh như đang bị rơi, đồng thời sẽ có dòng thông báo ứng với nội dung là "*Game Over*". Các đối tượng khác ngoài đối tượng *Bird* đều sẽ bị ẩn đi (*Visible = false*).



5. High Score:

- Chương trình sẽ có một *File* để lưu các giá trị *Score* của người chơi đạt được. Nếu có người chơi đạt được giá trị *Score* cao hơn giá trị *High Score* người chơi trước được lưu trước đó, sẽ có thông báo với nội dung chúc mừng *High Score* mới và đồng thời thông tin người chơi đạt *High Score* đó sẽ được cập nhật và hiển thị tại phần *High Score* của *Form 2*.

III – Link source code của nhóm:

https://github.com/LucasTran-tq/FlappyBirdPremium

CHƯƠNG V: NHẬN XÉT VÀ CẢI TIẾN

I – Nhận xét

* Những khó khăn gặp phải trong quá trình thực hiện đồ án:

- Chủ yếu là làm việc độc lập với nhau do hạn chế về mặt thời gian trống để tổ chức các buổi làm việc chung. Bên cạnh đó, ảnh hưởng của dịch Covid-19 cũng là một trong những nguyên nhân gây cản trở.

- Ban đầu, hầu hết các đối tượng gán hình ảnh trong chương trình đều được khởi tạo dưới dạng *PictureBox*. Vì có nhiều đối tượng, các tính năng tương ứng với từng đối tượng cùng tham gia vào quá trình chạy chương trình dẫn đến tình trạng *giật*, *lag* khi chơi của người dùng.

Nguyên nhân chính:

+ ***Bộ nhớ CPU hoạt động quá tải:*** do mỗi lần các đối tượng mang hình ảnh xuất hiện, chương trình sẽ lại nạp các thuộc tính của đối tượng đó một lần nữa vào chương trình.

+ **Tài nguyên bị sử dụng lãng phí:** do một số biến được khởi tạo nhưng không tham gia vào quá trình chơi.

- Việc cải thiện độ mượt mà khi trải nghiệm trò chơi trên nền tảng Windows Form còn nhiều hạn chế. Chỉ mới dừng lại ở mức *70 – 80%* sự mượt mà so với yêu cầu nhóm đặt ra.

- Vì tất cả thành viên trong nhóm đều chỉ mới học môn **Lập trình trực quan** mà chưa học qua các môn liên quan đến lập trình game nên khả năng lập trình, xây dựng và phát triển trò chơi **Flappy Bird** còn gặp nhiều hạn chế, chưa thực sự đạt đến sự hoàn thiện của các trò chơi đang hiện thành.

- Tính năng xoay (*rotate*) với khả năng hiện tại chưa thể xây dựng và phát triển, sẽ sớm được hoàn thiện trong thời gian sắp tới.

II – Cải tiến

- Để tăng tính ổn định, hiệu suất sử dụng CPU, tài nguyên một cách hiệu quả và tối ưu, thay vì sử dụng các *PictureBox* để khởi tạo cho tất cả các đối tượng mang hình ảnh, nhóm chúng em sử dụng hàm Draw trong sự kiện *Paint* của *Form* để chọn những đối tượng phù hợp với việc chạy trên ảnh tĩnh bình

thường, nên không chiếm lấy dung lượng và tài nguyên một cách bừa bãi. Cụ thể là chỉ cần *Bitmap* và vị trí của *Bitmap*.

- Luôn liên tục cập nhật các tính năng mới với các ý tưởng có thể thực thi trên *Windows Form*.

- Phát hiện, xử lý các lỗi *bug* chưa thực sự xét điều kiện sâu sát trong chương trình để ngày càng cải thiện sự trải nghiệm của người chơi hơn.

- Vẫn sẽ tiếp tục cập nhật trong thời gian tới ứng với việc đã tìm hiểu các nguồn tài liệu tham khảo khác để tăng vốn kiến thức lập trình và phát triển game hiện có.

TÀI LIỆU THAM KHẢO (ví dụ nếu có)

1. Flappy Bird phiên bản gốc:

https://www.mooict.com/create-flappy-bird-game-in-visual-studio-using-c/

2. Flappy Bird phiên bản nâng cao:

https://www.mooict.com/wpf-c-create-a-flappy-bird-game-in-visual-studio/

3. Guna2 with Windows Form:

https://www.youtube.com/watch?v=OKdGK8zr0k4

4. PictureBox with animated Gif in Windows Application:

https://stackoverflow.com/questions/13485477/can-a-picturebox-show-animated-gif-in-windows-application

5. Create an animated Gif:

https://ezgif.com/maker

6. Sound in Windows Application by Windows Media Player:

https://docs.microsoft.com/en-us/windows/win32/wmp/embedding-the-windows-media-player-control-in-a-c--solution?fbclid=IwAR2AvCCL5PXh6d58aw7A4gz1hygds9mL57WaEPGwH4_2v-DkC4p0zuEjn64

7. Image for background

https://stock.adobe.com/?fbclid=IwAR3nmfXWvsxrvY2ebknrSOzVoB4WZ0tNozDFAUPwKsd0N_dskbbey95ZCec