BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Báo cáo đồ án

Xây dựng game Tower Defense

Giảng viên hướng dẫn: Trần Anh Duy

Nhóm thực hiện: Nhóm 4

Lớp: 20CTT1TN2

Thành viên:

Nguyễn Anh Khoa – 20120118

Huỳnh Thiết Gia – 20120070

Trần Kiều Minh Lâm - 20120018

Báo cáo đồ án Tower Defense

Mục lục

I) Lời cảm ơn:

Mở đầu, Nhóm xin được gửi lời cảm ơn đến thầy Trần Duy Anh đã hỗ trợ nhiệt tình nhóm trong quá trình làm đồ án và đưa ra những lời khuyên kịp thời. Cảm ơn thầy Duy Anh và thầy Minh Huy đã truyền đạt kiến thức cần thiết để nhóm hoàn thành nhiệm vụ.

II) Giới thiệu đồ án:

Tower defense là tựa game chiến thuật phòng thủ khá quen thuộc với các game thủ. Đây là tựa game sở hữu lối chơi đơn giản, giao diện đẹp mắt, có tính giải trí cao, phù hợp với mọi lứa tuổi.

1) Hướng dẫn game:

a) Màn hình khởi đầu(menu):

Gồm có 2 nút

- Start: Khởi đầu màn chơi mới
- Quit: Thoát khỏi trò chơi

b) Giao diện chính trò chơi:

Hiển thị màn hình đầu tiên là 1 bản đồ trống với 2 phần thảm cỏ và đường đi. Quái vật sẽ di chuyển từ đầu này tới đầu kia của con đường, và chúng có sức tấn công cũng như lượng máu hoàn toàn khác nhau. Nhiệm vụ của người chơi là phải đặt các cỗ máy côn trùng giúp bảo vệ tòa thành nằm ở cuối con đường.

Sử dụng phím và chuột để thao tác:

- Để đặt máy ta nhấn phím từ 1 tới 5 để chọn thao tác, mỗi phím tương ứng với 1 loại quái vật. Khi ấn phím vào thì biểu tượng tại đó sẽ sáng lên và ta tiến

hành dùng chuột để đặt máy ở vị trí mong muốn, lưu ý ta không thể đặt trùng vị trí cỗ máy khác và trên đường đi của quái vật.

- Mỗi cỗ máy sẽ có hình dạng khác nhau và sát thương cũng khác nhau

- Ngoài ra, game còn có những vật phẩm hỗ trợ và ta có thể sử dụng nhờ ấn phím 5(đóng băng toàn bộ quái vật) và phím 6(tạo bom nổ toàn bộ quái trên màn hình hiện tại)

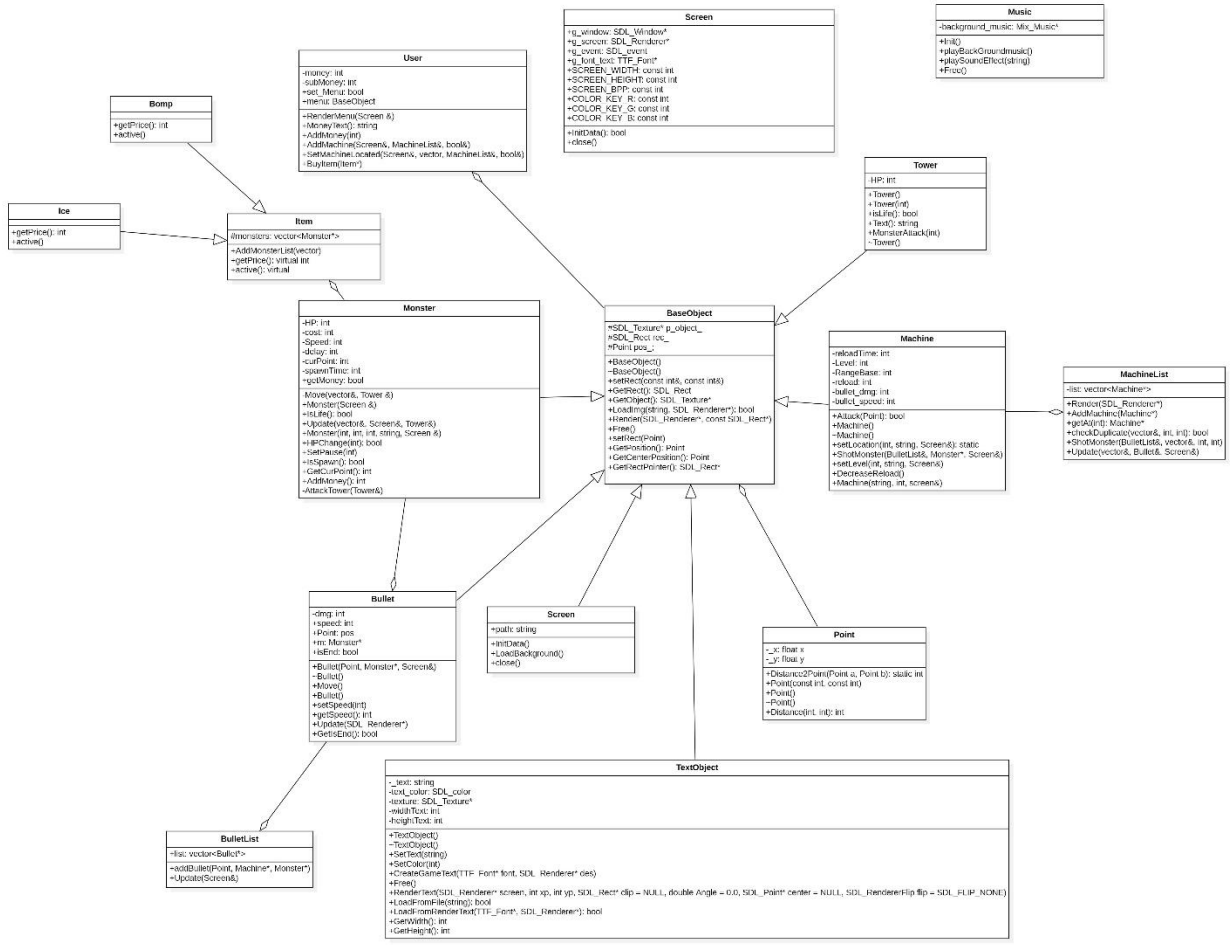
- Cả quá trình đặt cỗ máy và sử dụng vật phẩm đều tiêu hao 1 số tiền theo quy định, người chơi cần cân nhắc đặt sao cho phù hợp. Khởi đầu game người chơi sẽ có một số tiền là 1000. Trong quá trình tiêu diệt quái sẽ nhận được thêm tiền

Trò chơi yêu cầu người chơi phải tư duy để đặt số lượng máy sao cho hợp với tổng tiền và chừa trống các vị trí trọng điểm để đặt loại máy cao cấp hơn

c) Kết thúc game:

Nếu tiêu diệt toàn bộ quái vật, người chơi sẽ chiến thắng và kết quả sẽ hiển thị trên màn hình. Trái lại, nếu máu của tower về 0, người chơi sẽ thất bại trong việc bảo vệ. Ấn R để khởi động 1 lượt chơi mới khi kết thúc trò chơi.

III) Thiết kế game:



1) **BaseObject và những đối tượng cơ bản:**

Đây là đối tượng cơ bản chung cho mọi đối tượng khác trong game, mục tiêu của đối tượng này là hiển thị và xác định vị trí của bất kì một đối tượng nào khác trong game. Hay nói cách khác, đây là đối tượng cha của đa số các đối tượng khác.

Trong class BaseObject có những thuộc tính sau:

- SDL_ texture: đối tượng để render ra màn hình
- SDL_Rect: xác định hình chữ nhật bao quanh đối tượng
- Point: lấy vị trí của đối tượng

Trong class BaseObject có những phương thức sau:

- BaseObject(): Hàm khởi tạo thông thường, các thông số bằng 0 hoặc NULL
- ~BaseObject(): Hàm hủy, gọi hàm free()
- LoadImg(): nhận vào filepath và một đối tượng màn hình. tải hình ảnh lên cho 1 đối tượng baseObject, tạo một cái surface sau đó tạo texture dựa trên surface đó. Sau đó gán cho khối rect của đối tượng giá trị chiều dài chiều rộng vừa thu được. Giải phóng surface
- Render(): nhận vaò 1 đối tượng renderer(đóng vai trò như bản vẽ), để ta thực hiện vẽ trên đó dựa theo kích cỡ hình chữ nhật của đối tượng. Sau đó copy vào trong đối tượng renderer

a) TextObject:

Đây là đối tượng giúp hiển thị chữ lên màn hình, nó kế thừa từ chính baseobject. Đối tượng gồm có những thuộc tính sau:

- _text: dạng string chứa text
- color: quy định màu chữ
- texture_: bản vẽ để hiển thị
- Widthtext và heightext: các thông số kích thước

Các phương thức của TextObject:

- TextObject(): Khởi tạo đối tượng text
- SetColor(): tùy thuộc vào màu mà người dùng đưa vào, color có kiểu là SDL_color nhận 3 số tương ứng với red, blue, green
- Phương thức LoadFromRenderText: cũng tương tự ở lớp baseobject, ở đây nó tạo ra một text_surface(một hình), sau đó nó tạo ra một đối tượng texture_ từ chính text_surface ở trên

- Phương thức renderText cũng giống như ở BaseObject.

c) Music:

Class đóng vai trò tạo ra nhạc trong game, gồm có thuộc tính nhạc nền có kiểu dữ liệu là Mix_Music. Về phương thức:

- Phương thức Init(): Khởi tạo nhạc nền sau khi mọi thứ trước đó đã khởi tạo xong. Sau đó gọi hàm Mix_OpenAudio: truyền vào tần số, format(ở đây là 16-bit có dấu), truyền vào kênh là đa hay đơn(2 ở đây đại diện cho stereo) và chunksize là kích thước cho mỗi mẫu output. Sau đó sẽ load nhạc theo file có sẵn

- Phương thức playbackgroundMusic() và playSoundEffect(): để chơi nhạc

- Phương thức Free(): giải phóng vùng nhớ

d) Screen:

Một đối tượng quan trọng trong chương trình. Đóng vai trò như màn hình hiển thị mọi thứ trong chương trình. Các thuộc tính:

- g_window: cửa sổ cho chương trình

- g_renderer: Lớp nền của màn hình

- g_event: Bắt sự kiện trong chương trình

- g_font_text: font chữ sử dụng trong màn hình

- các thuộc tính hằng số khác như chiều cao màn hình, chiều rộng, BPP(Bits_per_pitxel) độ phân giải màn hình, các màu Red, green, blue

Các phương thức sử dụng:

- InitData(): Khởi tạo trên toàn màn hình: Khởi tạo SDL_INIT_VIDEO. SDL_SetHint nó thực hiện lọc tuyến tính và tạo ra một cửa sổ màn hình. Sau đó nó

tiếp tục tạo ra một màn vẽ nhờ hàm SDL_CreateRenderer, sau đó tiếp tục vẽ màu lên trên cho màn vẽ. Việc kế tiếp là sẽ khởi tạo font chữ để sử dụng trên màn hình

- close(): Đóng màn hình và tiến hành hủy màn vẽ trước(do màn vẽ tạo ra sau) sau đó hủy cửa sổ và thoát.

e) ClassPoint:

Nhận vào 2 giá trị x,y làm thuộc tính, gồm có phương thức khởi tạo và tính khoảng cách giữa 2 điểm

2) Machine:

a) Lớp đạn bắn:

Lớp được sử dụng để cỗ máy tấn công quái vật, được kế thừa từ BaseObject

Các thuộc tính

- Dmg: tượng trưng cho sát thương của đạn
- Speed: tốc độ của viên đạn
- Pos: Vị trí của viên đạn
- M: monster là đối tượng của đạn
- isEnd: kiểm tra xem viện đạn đã nổ chưa

Các thuộc tính:

- Bullet(): Khởi tạo bằng cách nhận vào điểm bắt đầu, mục tiêu và màn hình hiển thị
- Move(): hàm di chuyển của đạn. Nó sẽ tiến hành tính khoảng cách giữa đạn và mục tiêu, nếu khoảng cách là 0 thì cho 2 tọa độ đạn trùng với quái. Ngược lại, đạn sẽ tiến dần phía quái vật bằng cách cộng thêm 1 khoảng dựa trên tốc độ ban

đầu. Quaí vật bị giảm một lượng máu đúng bằng sát thương của đạn, viên đạn cũng đã nổ nên isEnd chuyển thành true.

- Update(): Cập nhật vị trí viên đạn, nhận vào là màn hình hiển thị, gồm ba thao tác, Move(), đặt lại vị trí đạn và xuất hình ảnh ra màn hình

b) Lớp băng đạn(Bullet List):

Chỉ gồm một thuộc tính duy nhất là 1 danh sách các con trỏ có kiểu là bullet. Gồm có 2 phương thức:

- addBullet: thêm đạn vào trong danh sách có sẵn

- Update: nhận vào màn hình chính. Nó sẽ duyệt qua toàn bộ từng thành phần trong băng đạn, kiểm tra xem viên đạn đó đã nổ chưa, nếu chưa thì tiếp tục update, nếu rồi thì xóa và giải phóng vùng nhớ cho viên đạn đó

c) Vật phẩm đặc biệt:

Class thực hiện tác động lên toàn bộ quái ở trạng thái hiện tại.

Thuộc tính:

- Chứa 1 list monster

Phương thức:

- GetPrice(): hàm ảo lấy giá tiiền của vật phẩm

- Active(): hàm ảo gọi hoạt động của vật phẩm

C1: Ice kế thừa từ lớp item và nó cho phép đóng băng toàn bộ quái vật trên màn hình. Hàm active của nó sẽ duyệt qua tất cả quái, quái nào đã được spawn rồi thì sẽ dừng lại

C2: Bomp kế thừa từ lớp item, cho phép tiêu diệt toàn bộ quái vật trên màn hình. Hàm active cũng duyệt toàn bộ mảng và kiểm tra tại mỗi vị trí quái đã xuất hiện chưa và còn sống hay không. Nếu thoải thì con đó sẽ bị tiêu diệt

d) Machine:

Class rất quan trọng trong game. Góp phần tạo nên chiến thắng của người chơi. Nó gồm những thuộc tính sau

- level: cấp của machine, ảnh hưởng đến tầm bắn và sát thương
- rangeBase: tầm bắn của machine
- reloadTime, reload: thời gian chờ giữa 2 lượt bắn
- bullet_dmg, bullet_speed: sát thương và tốc độ đạn nạp vào

Các phương thức của đối tượng:

- Machine(): khởi tạo gồm có 2 phương thức cơ bản. Một phương thức không có tham số và một phương thức có tham số. Một phương thức khởi tạo khác nhận vào filepath(tên hình ảnh), level và màn hình hiển thị
- ~Machine(): để hủy machine
- SetLevel: bổ sung thông tin level và filepath
- ShotMonster(): nhận vào đối tượng là 1 băng đạn, 1 đối tượng quái vật và màn hình hiển thị. Chương trình tiến hành kiểm tra khoảng cách giữa 2 điểm quái vật và vị trí machine hiện tại, sau đó tiến hành giảm thời gian chờ đi. Nếu reloadTime <= 0, quái vật còn sống và quái ở trong tầm bắn của machine thì cỗ máy mới tiến hành bắn. Khi đó ta gán lại giá trị reload bằng với reloadTime. Qúa trình bắn của đối tượng diễn ra khá đơn giản, chỉ cần thêm 1 viên đạn vào bullet list, sau đó mọi công việc còn lại thuộc phần nhiệm vụ của lớp bullet.

e) MachineList:

Áp dụng mẫu singleton để khởi tạo, nhằm khi khởi tạo chỉ tạo ra một list machine duy nhất, đa phần thì các thao tác qua machine đều thông qua một đối tượng trung gian là machineList, nó cũng có hơi hướng giống với mediator. Các thuộc tính của đối tượng:

- Một list machine chứa các machine:
- machineCount: Tượng trưng cho số máy đã sử dụng

Các phương thức của đối tượng:

- AdddMachine: Thêm 1 machine vào trong danh sách
- Render: Duyệt qua tất cả các đối tượng và render từng đối tượng đó ra màn hình
- GetAt: lấy đối tượng theo index trong danh sách
- CheckDuplicate: Hàm này sẽ thực hiện 2 công việc: cho phép người dùng đặt máy và kiểm tra thử vị trí đó có thể đặt được hay không. Đầu tiên, chương trình sẽ kiểm tra vị trí đó có trùng đường đi(thông qua file txt đã lưu bên ngoài) và với vị trí cỗ máy đã có sẵn hay không(đối chiếu với các máy đã đặt trong list) bằng hàm SDL_HandIntersection có sẵn của SDL. Nếu có nó chỉnh tọa độ cỗ máy tại điểm đó và trả về true, nếu không trả về false.
- ShotMonster(): Nhận vào 1 băng đạn, 1 list quái vật và màn hình hiển thị. Duyệt qua từng phần tử trong list Machine(từ đầu đến machineCount vì chỉ xét các máy đã đặt). Với mỗi cỗ máy duyệt toàn bộ danh sách quái vật và thực hiện bắn quái vật đó thông qua hàm shotMonster ở lớp Monster
- Update: Hàm này gọi 2 hàm liên tục là render cả list machine ra màn hình và thực hiện bắn thông qua hàm shotMonster() nằm ở cùng lớp.

3) Monster:

Đối tượng tấn công người chơi và đây cũng là đối tượng người chơi cần phải tiêu diệt. Các thuộc tính của đối tượng

- HP: máu của đối tượng
- speed: tốc độ di chuyển của quái vật
- curPoint: vị trí tính theo số điểm đã đi qua
- spawnTime: Thời gian chờ giữa 2 lần quái xuất hiện
- cost: số Tiền khi ta thu được một con quái
- getMoney: Kiểm tra xem ta có thể nhận tiền từ quái hay chưa
- delay: Thời gian chờ giữa 2 lần di chuyển.

Các phương thức đối tượng:

- Monster(): Khởi tạo một con quái vật, có hai loại phương thức khởi tạo là phương thức chỉ nhận vào màn hình và phương thức nhận đầy đủ thông số của quái vật. Trong các phương thức này lần lượt gán các giá trị cho các thuộc tính và tiến hành loadImg hình ảnh quái vật lên màn
- Move: Hàm di chuyển của quái vật, một lần gọi hàm này nếu giá trị delay > 0(quái vừa mới di chuyển thì ta giảm delay xuống 1 đơn vị rồi thoát), nếu delay <= 0 thì quái vật sẽ di chuyển tới điểm tiếp theo(vị trí của điểm đó được xác định bằng curPoint). Ngoài ra, nếu quái vật đến cuối đường thì sẽ tiến hành tấn công vào và làm giảm máu Tower và HP của monster về 0
- Update: Kiểm tra thử máu của monster có <=0 (đã chết) hay chưa, kiểm tra spawnTime có > 0 hay không, nếu còn lớn hơn thì giảm đi một đơn vị rồi trả về.

Nếu thỏa mãn spawnTime <= 0 và HP > 0 thì tiến hành di chuyển và render ra màn hình

- HPChange: Hàm này với mục tiêu là giảm máu của monster(nó sẽ trừ lượng máu hiện tại đi lượng damage truyền vào) nếu lượng máu <= 0 thì máu của monster trở về 0 và biến getMoney sẽ trả về true, khi đó ta có thể nhận tiền từ quái

- AttackTower: Hàm giúp quái tấn công trụ khi nó tới gần, gọi hai hàm là MonsterAttack và HPChange

- AddMoney: Kiểm tra xem getMoney đúng hay là sai, nếu đúng trả về 1 lượng tiền bằng với giá trị của quái vật

- IsLife: Để kiểm tra quái vật còn sống hay đã chết

- setPause: Dừng quái vật trong khoảng thời gian cho trước

- IsSpawn: kiểm tra quái vật đã xuất hiện chưa

- GetCurPoint: lấy tọa độ vị trí hiện tại

4) Các hàm chuyên dụng cho chế độ Dev:

- savePath: Hàm dùng để ghi vào trong file toàn bộ giá trị trong vector<Point> đầu vào

- readPath: Hàm dùng để đọc tọa độ các điểm từ fiile

getPath: hàm nhận vào một vector tọa độ các điểm của đường đi, các điểm đặt machine, và một vector chứa các hình chữ nhật.

- setPath: hàm dùng để tạo ra file path, tọa độ các điểm được lấy bằng cách kiểm tra event có di chuyển chuột hoặc nhấn chuột

5) Tower:

Lớp quyết định trạng thái thắng thua của trò chơi. Chỉ có một thuộc tính duy nhất là lượng máu. Phương thức:

- Tower(): Khởi tạo lượng máu mặc định là 1000, ngoài ra có thể điều chỉnh bằng cách truyền thêm thông số đầu vào
- isLife(): Kiểm tra trụ còn máu hay không
- Text(): Trả về string để hiển thị máu trên màn hình
- MonsterAttack(): Giảm máu của trụ theo lượng dmg cho trước

6) Lớp User:

Tượng trưng cho người dùng. Đại diện cho người chơi game. Có các thuộc tính như sau:

- Money: Tượng trưng cho số tiền người đó sở hữu
- subMoney: Số tiền người dùng phải trả khi mua machine hoặc item bất kì
- set_menu = hiển thị bảng chọn trên màn hình
- menu: Tượng trưng cho bản chọn trên màn hình

Thuộc tính:

- MoneyText(): trả về chuỗi kí tự để hiển thị tiền
- AddMoney(): Thêm tiền vào trong tài khoản của người đó, đầu vào là số tiền cần thêm
- LoadMenu(): Load hình ảnh menu để hiển thị ra màn hình
- RenderMenu(): Bình thường sẽ hiển thị hình mặc định là Use0.png(toàn bộ các đối tượng đều có màu xám, kiểm tra bằng cách trạng thái của set menu bằng false), và nếu như ta có sử dụng phím để chọn trước đó, hàm loadImg

đã được gọi rồi. Sau đó chương trình sẽ set Tọa độ cho hình chữ nhật và render ra màn hình

- AddMachine: Hàm này sẽ thực hiện khi biến is_set_machine nhận giá trị false(kiểm tra trong main), ta sẽ kiểm tra giá trị nút ấn của bàn phím, và với mỗi giá trí của phím, ta sẽ tiến hành làm sáng menu tại đó lên, kiểm tra số tiền có thỏa mãn. Nếu số tiền thỏa mãn, ta sẽ set level cho máy và gán biến subMoney bằng số tiền cần chi trả để mua máy đó. Ta chuyển trạng thái is_set_machine về true
- Hàm SetMachineLocated: sẽ thiết lập tọa độ cho machine, nó sẽ được gọi khi is_set_machine nhận giá trị của true. Hàm này sẽ lấy tọa độ của chuột rồi đem đi kiểm tra có trùng với máy đã đặt trước đó hay không thông qua hàm checkDuplicate. Nếu có thể đặt được, số tiền của người chơi sẽ trừ đi một lượng subMoney, và subMoney sẽ trả về 0, biến is_set_machine cũng trở lại nhận giá trị false.
- BuyItem(): Hàm này để user sẽ mua những vật phẩm đặc biệt trong lớp Item, thông qua tính đa hình có thể active item tương ứng, sau đó cũng trừ đi một số tiền ứng với item đó.

7) Thao tác trong main:

- LoadBackground: thực hiện loadImage ra ngoài màn hình.
- Khởi tạo các giá trị cần thiết trong main: Time, machineCount, musicControl
- Đầu tiên gọi hàm displayMenu(): để hiển thị menu ra màn hình
- Khởi tạo nhạc bằng MusicControl: Cấp phát, gọi hàm Init() và playBackground()

- Khai báo một đối tượng MainScreen, g_background. Gọi hàm InitData() của MainScreen và thực hiện LoadBackground

- Khai báo TextObject: Print_text: số máu của trụ và Money_text: Số tiền của người chơi, sau đó thiết lập màu sắc cho text

- Khởi tạo biến is_quit để kiểm tra đúng sai

- Tạo ra một User chơi game và một băng đạn. Khởi tạo 1 list Machine gồm 100 máy(số máy đặt được tối đa). Sau đó Tạo ra một tower ở vị trí cuối con đường

- PathInit: Khởi tạo các list Point từ file txt có sẵn từ chế độ dev

- Khởi tạo một danh sách Monster với tốc độ và máu khác nhau, thời gian hồi cũng khác nhau

- Khởi tạo 2 item đặc biệt và add đối tượng bị ảnh hưởng bởi nó là MonsterList vào

Tạo một vòng lặp while chạy khi is_quit nhận giá trị false. Trong vòng lặp đó ta thực hiện những thao tác sau:

- Kiểm tra biến allMonsterDied nếu nhận giá trị true thì game thắng, load ảnh nền chiến thắng lên màn hình
- Kiểm tra điều kiện của sự kiện, nếu ấn vào nút thoát thì ta thoát chương trình và nếu ấn R thì trò chơi sẽ khởi động lại
- Kiểm tra trụ có còn máu hay không, nếu không đồng nghĩa với thua cuộc, ta cũng cho phép người dùng ấn R để khởi động lại game
- Bắt sự kiện trong trò chơi chính: chương trình sẽ thoát nếu ấn dấu x ở góc phải trên. Sau đó chương trình kiểm tra nếu biến is_set_machine nhận giá trị true, ta sẽ đặt nó tại một tọa độ theo hàm setMachineLocated ở lớp User.

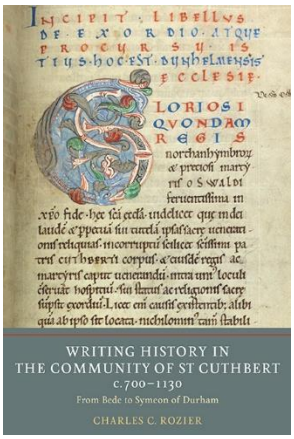
Đồng thời nếu is_set_machine nhận giá trị false và người chơi nhấn một phím nào từ 1->4 thì sẽ tiến hành đặt các cỗ máy. Nếu sự kiện ấn phím 5,6 thì ta tiến hành mua vật phẩm đặc biệt và sử dụng

- Trong gameProcess, ta tiến hành vẽ lên màn hình và render g_screen lên đó
- Trong Money và TowerProcess, ta chỉ có nhiệm vụ là hiển thị thông tin lên màn hình
- Cứ mỗi frame ta update liên tục machineList và bulletList. Trong quá trình Process Monster, ta sẽ lần lượt Update và nhận tiền từ từng monster trong list, sau đó đếm số quái con sống trong danh sách. Nếu danh sách không còn quái thì allMonsterDied chuyển thành true
- Sau đó là ta tăng thời gian lên và delay 1 khoảng để đến frame tiếp theo
  Kết thúc vòng while, ta sẽ tiến hành giải phóng cho g_backround, mainscreen và musicControl

8) Menu:

Vì tên game là Tower (trụ/tháp/thành) Defense và các máy bắn ra đạn nhìn giống những viên đại bác nên nhóm lấy hình tượng thành trì thời kỳ trung cổ châu u làm phong cách chủ đạo cho Intro.

Sử dụng font được thiết kế theo chữ viết cùng thời đặt trong lá cờ tạo ra một logo cách điệu nhưng vẫn gột tả được vẻ hào hùng của trận chiến.



Vì trò chơi được thiết kế với độ phân giải cố định nên ta có thể thay hộp thoại chữ bằng hình ảnh, tái sử dụng lại code hiển thị ảnh nền giúp giảm thiểu sự phức tạp của code. Thêm code điều kiện kiểm tra nếu tọa độ con trỏ nằm trong hộp thoại (tọa độ con trỏ lớn hơn góc trái trên và nhỏ hơn góc phải dưới) thì đổi qua phiên bản hộp thoại được chọn.

9) Design Pattern

- Sử dụng singleton để khởi tạo một đối tượng MachineList duy nhất, khi thay đổi chỉ cần thay đổi những Machine trong list mà thôi

- Sử dụng Observer để khi ta gọi item thì item sẽ thông báo đến tất cả monster sẽ dừng lại hoặc sẽ tự động chết theo yêu cầu của item. Thì những item hiện tại trên màn hình chính là item nằm trong list, khi có quái vật nào bắt đầu được spawn ra màn hình thì nó sẽ được tính là bị tác động và gửi thông báo cho quái đó tự thay

đổi(chết hoặc đóng băng). Nếu quái đó chết hoặc chưa xuất hiện thì sẽ không bị ảnh hưởng.

IV) Các công nghệ sử dụng:

1) Về tài nguyên:

- Asset Game: https://craftpix.net/freebies/
- Hình ảnh từ game: Liên minh huyền thoại

2) Về thư viện sử dụng:

- Bộ thư viên SDL2, code trên nền tảng C++ sử dụng kĩ thuật lập trình hướng đối tượng
- Các mẫu design Pattern sử dụng như Singleton (Khởi tạo machine), Observer (User khi dùng item thì tác động lại tất cả các đối tượng monster)

3) Nguồn tham khảo:

- Kênh youtube:
- https://www.youtube.com/watch?v=QQzAHcojEKg
  https://www.youtube.com/watch?v=k1JGvJU707k

V) Phân chia công việc:

Họ tên và MSSV	Phần công việc	Tổng
Nguyễn Anh Khoa - 20120118	Phần Khởi tạo game, và thiết kế game: machine, viết báo cáo, UML	35%
Trần Kiều Minh Lâm – 20120018	Phần thiết kế game: Monster, trạng thái thắng thua, xây dựng lớp đạn bắn, nhạc game	35%
Huỳnh Thiết Gia – 20120070	Menu game, design ảnh, hỗ trợ viết báo cáo, state lose và win của game	30%

Tự chấm điểm hoàn thành đồ án 9.25/11