TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN
MÔN: **LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

HƯỚNG DẪN ĐỒ ÁN TRÒ CHƠI
BĂNG QUA ĐƯỜNG

TP.HCM, ngày 09 tháng 09 năm 2017

MỤC LỤC

1 Giới thiệu

Trong phần đồ án này ta sẽ phối hợp các kĩ thuật, cấu trúc dữ liệu cơ bản và kiến thức lập trình hướng đối tượng để xây dựng một trò chơi băng qua đường (road crossing).

Để thực hiện được đồ án này ta cần các kiến thức cơ bản như: xử lý tập tin, tiểu trình, handle, các cấu trúc dữ liệu cơ bản và kiến thức lập trình hướng đối tượng…

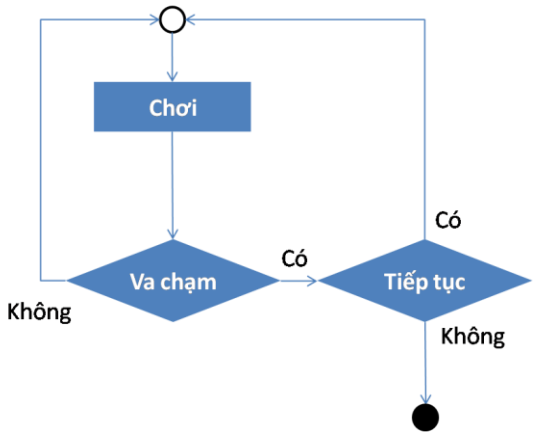
Phần hướng dẫn giúp sinh viên xây dựng trò chơi ở mức độ cơ bản, các em tự nghiên cứu để hoàn thiện một cách tốt nhất có thể.

2 Kịch bản trò chơi

Lúc đầu khi vào game sẽ xuất hiện các xe và thú chạy qua lại và một kí tự "Y" đại diện cho người qua đường, người chơi sử dụng các phím 'W', 'A', 'S', 'D' để điều chỉnh hướng di chuyển của người qua đường và cố gắng tránh các xe và thú.

Khi "Y" va chạm các xe hay thú thì chương trình thông báo yêu cầu người chơi chọn phím 'y' nếu muốn tiếp tục (chương trình sẽ thiết lập lại dữ liệu trò chơi lại như lúc ban đầu) hoặc chọn 'bất kì phím nào' nếu muốn thoát trò chơi.

Khi "Y" đi qua được hết các xe và thú thì sẽ lên cấp kế tiếp, độ khó của trò chơi chính là số lượng xe và thú tham gia di chuyển trên đường (Vị trí của "Y" mới sẽ xuất hiện trở lại khi lên cấp). Khi lên cấp tối đa nào đó thì dữ liệu sẽ khởi động lại như lúc ban đầu.



Hình 1: Sơ đồ kịch bản trò chơi

3 Các kĩ thuật hỗ trợ

Trong phần này ta sẽ lần lượt được giới thiệu các kĩ thuật hỗ trợ quá trình phát triển trò chơi (sinh viên tự thiết kế mẫu phù hợp trong quá trình làm đồ án).

3.1 Cố định màn hình chính

Trong quá trình chơi trò chơi, người dùng có thể thay đổi kích thước cửa sổ, điều này sẽ làm cho các đối tượng thay đổi kích thước gây khó khăn trong quá trình tính toán. Ta nên cố định màn hình trước khi thực hiện các tính toán.

```
Dòng
1  void FixConsoleWindow() {
2     HWND consoleWindow = GetConsoleWindow();
3     LONG style = GetWindowLong(consoleWindow, GWL_STYLE);
4     style = style & ~(WS_MAXIMIZEBOX) & ~(WS_THICKFRAME);
5     SetWindowLong(consoleWindow, GWL_STYLE, style);
6  }
```

Trong đoạn mã trên, kiểu HWND là một con trỏ trỏ tới chính cửa sổ Console. Để làm việc với các đối tượng đồ họa này, ta cần có những kiểu như thế. Cờ GWL_STYLE được xem là dấu hiệu để hàm GetWindowLong lấy các đặc tính mà cửa sổ Console đang có. Kết quả trả về của hàm GetWindowLong là một số kiểu long, ta sẽ hiệu chỉnh tại dòng số 4. Ý nghĩa là để làm mờ đi nút maximize và không cho người dùng thay đổi kích thước cửa sổ hiện hành. Sau khi đã hiệu chỉnh xong, ta dùng hàm SetWindowLong để gán kết quả hiệu chỉnh trở lại. Ta có thể thử nghiệm hàm trên và tự xem kết quả.

3.2 Thiết lập vị trí cho con trỏ màn hình

Trong trò chơi sẽ có rất nhiều vị trí mà ta muốn in tại đó, vì vậy ta cần có khả năng di chuyển tới tất cả các vị trí trong màn hình console.

```
Dòng
1  void GotoXY(int x, int y) {
2     COORD coord;
3     coord.X = x;
4     coord.Y = y;
5     SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
6  }
```

Trong đoạn mã này ta sử dụng struct _COORD (COORD), đây là một cấu trúc dành xử lý cho tọa độ trên màn hình console. Ta gán hoành độ và tung độ cho biến coord sau đó thiết lập vị trí lên màn hình bằng hàm SetConsoleCursorPosition. Lưu ý: hàm này cần một đối tượng chính là màn hình console (màn hình đen), vì vậy ta cũng cần có một con

trỏ trỏ tới đối tượng này (HANDLE thực chất là void*). Ta có được bằng cách gọi hàm GetStdHandle với tham số là một cờ STD_OUTPUT_HANDLE.

3.3 Kĩ thuật đa tiểu trình

Tiếp theo ta xem xét cơ chế hai tiểu trình chạy song song gồm một tiểu trình main và một tiểu trình phụ.

Dòng	
1	
2	void exitGame(thread* t){
3	system("cls");
4	IS_RUNNING = false;
5	t->join();
6	}
7	void ThreadFunc1(){
8	while(IS_RUNNING){
9	//…thực hiện in ra màn hình console những đối tượng trong trò chơi
10	}
11	void main(){
12	//…
13	thread t1(ThreadFunc1); //Tạo một thread phụ chạy song song với thread main
14	while(1){
15	int temp = toupper(getch());
16	//…
17	if(temp == 27) {// người dùng muốn thoát
18	exitGame(&t1); return;
19	}
20	//…
21	}

Trong đoạn mã trên ta thấy hàm 'main' sẽ tạo ra một thread con chạy song song với mình. Trong hàm ThreadFunc1 (đại diện cho thread con này) sẽ có một vòng lặp chạy với điều kiện biến IS_RUNNING còn là true (vì vậy lúc đầu biến này phải có giá trị true). Trong quá trình 'main' chạy nếu người dùng nhấn phím 'ESC' thì sẽ gọi hàm 'exitGame'. Trong hàm 'exitGame' ta sẽ gán giá trị false cho biến IS_RUNNING để hàm ThreadFunc1 dừng lại, đồng thời ta cho tiểu trình 't' join với hàm 'main' (Vì theo quy định tiểu trình 'main' phải kết thúc sau các tiểu trình con).

3.4 Các lớp trong trò chơi

Tiếp theo ta sẽ giới thiệu các đối tượng cần thiết trong trò chơi băng qua đường cũng như sự tác động qua lại giữa các đối tượng.

3.4.1 Lớp CPEOPLE

Trong trò chơi này sẽ có một đối tượng người di chuyển qua các làn xe chạy. Có thể dùng kí tự 'Y' đại diện hoặc tự thiết kế một hình mẫu nào đó tùy ý. Cơ bản lớp người có các thông tin cơ bản sau:

```
class CPEOPLE{
  int mX, mY;
  bool mState; //Trạng thái sống chết
  public:
    CPEOPLE();
    void Up(int);
    void Left(int);
    void Right(int);
    void Down(int);
    bool isImpact(const CVEHICLE*&);
    bool isImpact(const CANIMAL*&);
    bool isFinish();
    bool isDead();
}
```

Trong đó các phương thức Up, Left, Right, Down cập nhật vị trí của đối tượng CPEOPLE, hai phương thức isImpact kiểm tra trường hợp khi chạm các đối tượng CVEHICLE và CANIMAIL, ví dụ nếu va chạm thì state phải là false (chết)... Ngoài các phương thức cơ bản đó ra, sinh viên có thể tự thiết kế thêm các thuộc tính và phương thức cần thiết khác.

3.4.2 Lớp CVEHICLE

Trong quá trình người băng qua đường, các xe sẽ chạy, về cơ bản lớp xe có các thông tin như sau

```
class CVEHICLE{
  int mX, mY;
  public:
    virtual void Move(int, int);
   //…
}
```

static int m_Amount;

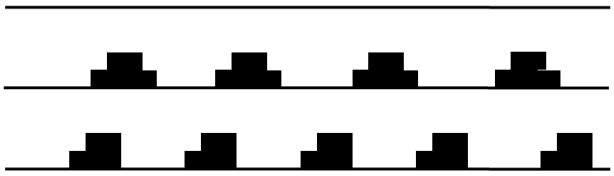
public:
static int getAmount();

Trong đó phương thức ảo Move sẽ thực hiện cập nhật vị trí mới của đối tượng CVEHICLE. Ngoài ra, sinh viên tự thiết kế thêm một số phương thức và thuộc tính khác. Lưu ý lớp CVEHICLE chỉ là lớp tổng quát, ta sẽ có hai lớp con là CTRUCK và CCAR

```
class CTRUCK: public CVEHICLE{
  public:
   //…
}
```

```
class CCAR: public CVEHICLE{
  public:
   //…
}
```

Sau đây là hình ảnh minh họa khi vẽ ra màn hình hai làn xe



3.4.3 Lớp CANIMAL

Tương tự như lớp CVEHICLE, ta sẽ có lớp CANIMAL với các thông tin cơ bản như sau

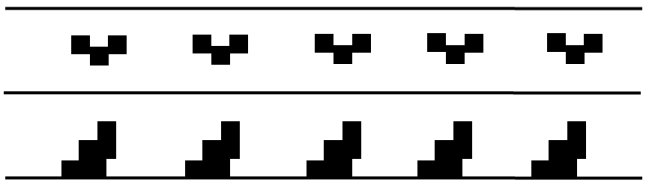
```
class CANIMAL{
  int mX, mY;
  public:
    virtual void Move(int, int);
    virtual void Tell();
}
```

Trong đó phương thức Move sẽ thực hiện cập nhật vị trí mới của đối tượng CANIMAL tùy vào từng loại, phương thức Tell() sẽ phát ra tiếng kêu với ứng với từng loài. Ngoài ra, sinh viên tự thiết kế thêm một số phương thức và thuộc tính khác. Lưu ý lớp CANIMAL chỉ là lớp tổng quát, ta sẽ có hai lớp con là CBIRD và CDINAUSOR

```
class CDINAUSOR: public CANIMAL{
  public:
   //…
}
```

```
class CBIRD: public CANIMAL{
  public:
   //…
}
```

Sau đây là hình ảnh minh họa in ra hai làn thú



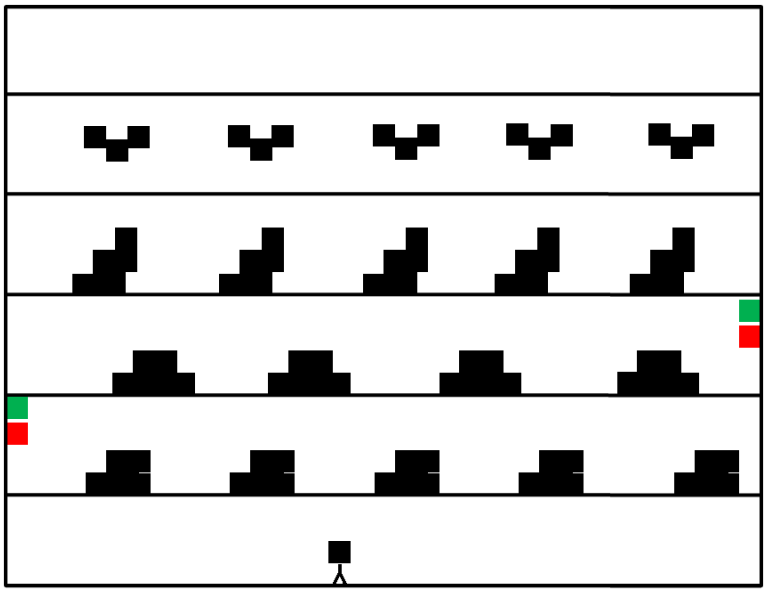
3.4.4 Lớp CGAME

Như vậy các đối tượng trong trò chơi đã đầy đủ. Cuối cùng ta có lớp CGAME, lớp này đóng vai trò trung tâm trò chơi sẽ điều phối toàn bộ các đối tượng trong trò chơi này. Lớp game bao gồm các thông tin quan trọng sau

Dòng	
1	class CGAME{
2	CTRUCK* axt;
3	CCAR* axh;
4	CDINAUSOR* akl;
5	CBIRD* ac;
6	CPEOPLE cn;
7	public:
8	CGAME(); //Chuẩn bị dữ liệu cho tất cả các đối tượng
9	void drawGame(); //Thực hiện vẽ trò chơi ra màn hình sau khi có dữ liệu
10	~CGAME(); // Hủy tài nguyên đã cấp phát
11	CPEOPLE getPeople();//Lấy thông tin người
12	CVEHICLE* getVehicle();//Lấy danh sách các xe
13	CANIMAL* getAnimal(); //Lấy danh sách các thú
14	void resetGame(); // Thực hiện thiết lập lại toàn bộ dữ liệu như lúc đầu
15	void exitGame(HANDLE); // Thực hiện thoát Thread
16	void startGame(); // Thực hiện bắt đầu vào trò chơi
17	void loadGame(istream); // Thực hiện tải lại trò chơi đã lưu
18	void saveGame(istream); // Thực hiện lưu lại dữ liệu trò chơi
19	void pauseGame(HANDLE); // Tạm dừng Thread
20	void resumeGame(HANDLE); //Quay lai Thread
21	void updatePosPeople(char); //Thực hiện điều khiển di chuyển của CPEOPLE
22	void updatePosVehicle(); //Thực hiện cho CTRUCK & CCAR di chuyển
23	void updatePosAnimal();//Thực hiện cho CDINAUSOR & CBIRD di chuyển
24	}

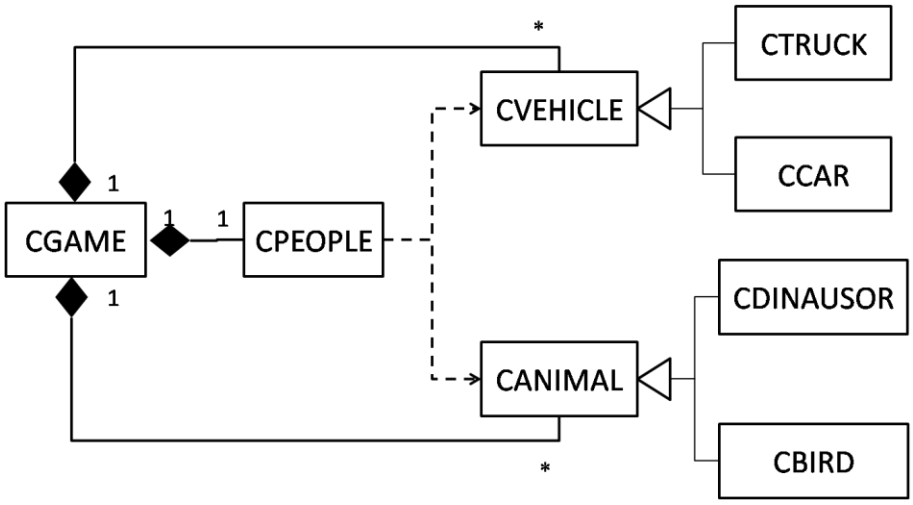
Bên cạnh đó, sinh viên có thể tự định nghĩa một vài hằng số (const) quy định các tham số trong trò chơi ví dụ MAX_LEVEL qui định số cấp tối đa hay MAX_BIRD qui định số chim bay tối đa…

Hình ảnh minh họa trò chơi hoàn chỉnh



3.4.5 Sơ đồ lớp trò chơi

Phần này cho ta cái nhìn tổng thể về các lớp tham gia trong trò chơi này



3.5 Đoạn mã giả minh họa

Phần này trình bày cách sử dụng trong hàm main với tiểu trình và các đối tượng trong trò chơi.

Dòng	
1	`//Các hằng số và biến toàn cục cần thiết`
2	`char MOVING;`
3	`CGAME cg;`
4	`void main()`
5	`{`

```
6    cg = new CGAME();
7    int temp;
8    FixConsoleWindow();
9    cg.startGame();
10   thread t1(SubThread);
11   while (1)
12   {
13     temp = toupper(getch());
14     if (!cg.getPeople().isDead())
15     {
16       if (temp == 27) {
17         cg.exitGame(t1.native_handle());
18         return;
19       }
20       else if (temp == 'P') {
21         cg.pauseGame(t1.native_handle());
22       }
23       else {
24         cg.resumeGame((HANDLE)t1.native_handle());
25         MOVING = temp; //Cập nhật bước di chuyển
26       }
27     }
28     else
29     {
30       if (temp == 'Y') cg.startGame();
31       else {
32         cg.exitGame(t1.native_handle());
33         return;
34       }
35     }
36   }
37 }
```

Ngoài ra phần tiểu trình chạy song song cũng quan trọng điều phối chính các đối tượng CANIMAL và CVEHICLE trên màn hình

```
Dòng
1    void SubThread()
2    {
3      while (IS_RUNNING) {
4        if (!cg.getPeople().isDead()) //Nếu người vẫn còn sống
5        {
6          cg.updatePosPeople(MOVING);//Cập nhật vị trí người theo thông tin từ main
```

```
7       }
8       MOVING = ' ';// Tạm khóa không cho di chuyển, chờ nhận phím từ hàm main
9       cg.updatePosVehicle();//Cập nhật vị trí xe
10      cg.updatePosAnimal(); //Cập nhật vị trí thú
11      cg.drawGame();
12      if (cg.getPeople().isImpact(cg.getVehicle() ||
                                    cg.getPeople().isImpact(cg.getAnimal())
13      {
14        // Xử lý khi đụng xe hay thú
15      }
16      if (cg.getPeople().isFinish()){
17          // Xử lý khi về đích
18      }
19      Sleep(100);
20    }
21  }
```

4 YÊU CẦU ĐỒ ÁN

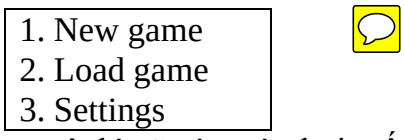
Trong phần hướng dẫn trên ta còn thiếu một vài chức năng cơ bản

4.1 Cài đặt chạy được giống kịch bản mô tả (3đ)

Xem lại mô tả kịch bản và cài đặt để trò chơi hoạt động giống mô tả, lưu ý: các hình vẽ xe tải, xe hơi ở cấp độ này chưa cần thiết, sinh viên có thể dùng kí tự nào đó đại diện.

4.2 Xây dựng thực đơn cho trò chơi khi vừa mới vào (1đ)

Xây dựng trình menu cho trò chơi, ví dụ như lúc đầu sẽ hiện lên bản



Khi người dùng chọn phần ‘New game’ thì sẽ vào trò chơi, nếu chọn phần ‘Load game’ hay ‘Settings’ thì tạm chưa xử lý và yêu cầu người dùng chọn lại.

4.3 Xử lý lưu/tải trò chơi (3đ)

Sinh viên bổ sung chức năng ‘Load game’ khi người dùng nhấn vào menu lúc đầu hoặc nhấn phím ‘T’ khi đang chơi. Lúc này chương trình tạm dừng và in ra dòng yêu cầu người dùng nhập đường dẫn tập tin đã lưu. Khi đó chương trình thiết lập dữ liệu và vào trò chơi.

Sinh viên bổ sung chức năng ‘Save game’ khi người dùng nhấn phím ‘L’ khi đang chơi. Lúc này chương trình tạm dừng in ra dòng yêu cầu người dùng nhập đường dẫn tập tin

cần lưu. Sau khi lưu xong thì chương trình tiếp tục chơi (có thể hỏi người chơi có muốn tiếp tục hay không).

Hướng dẫn: Sinh viên tự tổ chức cấu trúc tập tin để lưu dữ liệu biến toàn cục trong chương trình

4.4 Xử lý tạm dừng các toa xe (2đ)

Trong hướng dẫn, các xe di chuyển liên tục, sinh viên hãy hiệu chỉnh lại sao cho mỗi toa xe đều có thể dừng lại trong một khoảng thời gian tùy ý để giúp trò chơi dễ chơi hơn khi lên cấp. Gợi ý: Có thể xây dựng lớp CTRAFFICLIGHT kết hợp với lớp CVEHICLE để tạm dừng khi tín hiệu đèn đỏ.

4.5 Xử lý hiệu ứng khi va chạm (0.5đ)

Khi người qua đường va chạm xe thì tạo hiệu ứng đơn giản minh họa việc va chạm, khi các con thú di chuyển có thể phát ra tiếng kêu...

4.6 Giao diện (0.5đ)

Cách thức bố trí các thông tin trò chơi 'hợp lý', có hình vẽ cụ thể của các xe, thú, người và đèn tín hiệu…