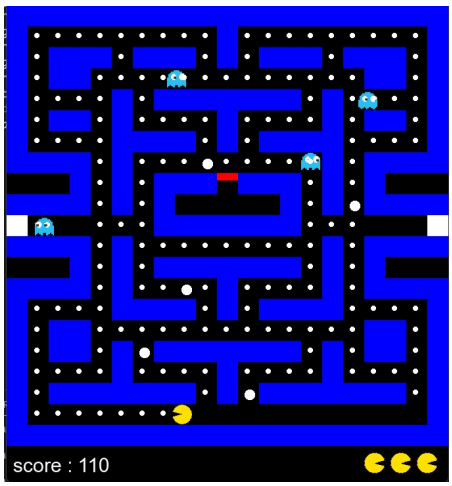
دانشکده فنی و مهندسی

برنامه سازی پیشرفته

گزارش کار

پروژه پایانی درس برنامه نویسی پیشرفته (pacman)



نام و نام خانوادگی : حسین حقیقت

شماره دانشجویی : 40012358010

استاد

دکتر مرتضی یوسف صنعتی

مرداد 1401

فهرست

معرفی بازی

بازی پک من (pac man) یک بازی قدیمی با ساختار آرکید میباشد که برای اولین بـار در سـال 1980 در کشور ژاپن توسط شرکت نامکو ساخته و معرفی شد[1] .

- در این بازی شما با نقش پکمن حضور دارید که هدف اصلی جمـع آوری امتیـاز و دریافـت بـالا ترین رکورد میباشد .

- روح ها در این بازی نقش دشمن شما را بازی کرده و هدف اصلیشان نابودی شما میباشد پس تا حد ممکن باید سعی کنید از آنها دور بمانید .

  همچنین خوراکی های متفاوتی در زمین دیده میشود که هرکدام امتیاز یا ویژگی های متفاوت خود را دارا میباشد .

  شما میتوانید با خوردن هر خوراکی عادی ده امتیاز مثبت دریافت کنید .

- خوراکی های ویژه (قدرتی) که اندازه بزرگ تری نسبت به خوراکی های عادی دارند علاوه بر 50 امتیازی که به شما میدهند ، میتوانند شما را برای مدت کوتاهی تبدیل به یک روح گیر کننـد . شما در این حالت که رنگ روح ها تغییر کرده میتوانید آنها را اسیر کرده و 200 امتیاز دریافـت کنید .

- میوه ها هم در هر نوبت از بازی دوبار و به مدت محدود در زمین ظاهر میشوند که با خوردن آنها امتیاز بالایی نصیب پکمن خواهد شد .

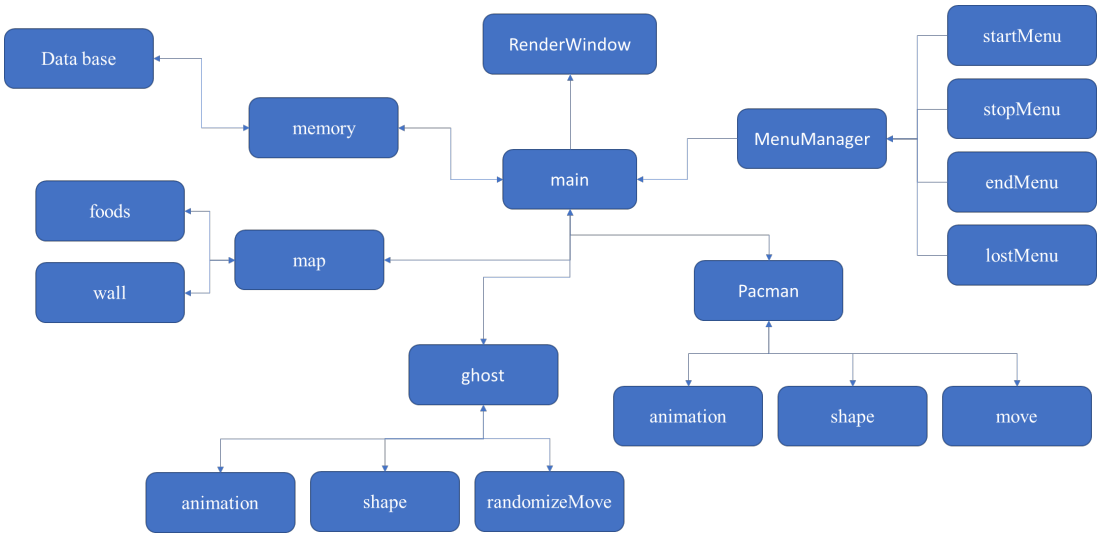
---

[1] fa.wikipedia.org

1 ساختار کلی پروژه

مرکزیت پروژه و مدیریت اصلی المان ها و حلقه اصلی بازی در فایل main.cpp رخ میدهد .

این بخش با هر یک از بخش های دیگر در ارتباط است و اطلاعاتی را رد و بدل میکند .



1.1 RenderWindow :

کلاس پنجره که یک کلاس از پیش ساخته شده از کتابخانه قدرتمند sfml است وظیفه طراحی رابط گرافیکی بازی را بر عهده دارد .

در ابتدای فایل اصلی یک شی از این کلاس میسازیم و در بخش های مختلف عناصر خودمان را داخل آن ثبت و به نمایش میگذاریم .

در هر بخش از برنامه که نیاز به این شی ساخته شده باشد آدرس همین شی ساخته شده را به عنوان مرجع دریافت و در آن تغییرات را انجام میدهیم .

1.2 MenuManager :

فضای نام MenuManager به شما این امکان را میدهد که در هر قسمت از پروژه تنها با فراخـانی یـک تابع ، منو مورد نظر را به کاربر نمایش دهید و نتیجه تصمیم گیری و گزینه انتخاب شده توسط کـاربر را از همان تابع دریافت کنید .

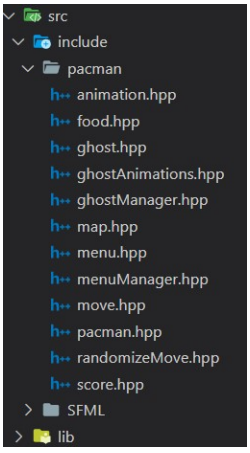
انواع منو در این فضای نامی عبارت اند از :

- startMenu : اولین منو که هنگام استارت بازی نمایش داده میشود
- stopMenu : این منو در زمان اجرای بازی با کلید (ESC) نمایش داده میشود
- endMenu : زمانی که کاربر یک مرحله را به اتمام میرساند این منو فراخانی میشود
- lostMenu : کاربر بعد هر باخت این منو را همراه با بیشترین رکـورد و همچنـین رکـورد خـود میتواند مشاهده کند

هر منو متشکل از لیبل ها و کلید هایی است که انتخاب های مختلفی را در اختیار کاربر قـرار میدهـد و در آخر انتخاب کاربر را بر میگرداند که توسعه دهنده با یک switch میتواند برای هر نتیجه برنامه خاص خود را اختصاص دهد .

1.3 کلاس های پایه

کلاس های پایه ، کلاس هایی هستند که در ابتدا ساخته شدند تا طراحی بـازی را مـاژولار و بـه مراتـب ساده تر کند . هرکدام از کلاس های پایه در فایلی با همـان نـام ذخیـره شـده است تا دسترسی به آن کلاس را راحت تر کند .



1.3.1 Map :

```
std::vector<sf::RectangleShape> blocks;
std::vector<fd::Food> foods;
sf::RectangleShape leftTeleport;
sf::RectangleShape rightTeleport;
sf::RectangleShape door;
sf::Time _time;
sf::Clock _clock;
int h_block;
int w_block;
int foodCounter = 0;
int visibleFoods = 0;
int specialFoodNumber = -1;
```

کلاس نقشه شامل دو وکتور میباشد . یکی از وکتور ها وظیفه نگهداری بلاک های دیوار و دیگری وظیفه نگهداری خوراکی های موجود در نقشه را دارد .دو بلاک برای دروازه های اطراف زمین که جابه جایی در آنها صورت میگیرد در کلاس نقشه قرار دارد . یک بلاک هم به عنوان در خانه ارواح ایجاد شده است . این کلاس تعداد خوراکی های خورده شده و همچنین خوراکی های فعال و باقی مانده در زمین ذخیره و نگهداری میکند .

متغیر specialFoodNumber در حالت عادی با منفی یک مقدار دهی شده است ولی پس از خوردن هفتاد خوراکی ، خوراکی ویژه (میوه) ای در صفحه ظاهر میشود . در آن زمان است که specialFoodNumber شمارنده خاص آن خوراکی را در خود ذخیره میکند .

```
Map(int h, int w);
void draw(sf::RenderWindow & window);
int accident(sf::RectangleShape & shape, bool ghost = false);
void setRandomFood(fd::Type t);
void checkSpecoalFood();
int getFoodCounter();
int getVisibleFoods();
void restart();
```

همچنین متد های این کلاس عبارت اند از :

- Draw : وظیفه ثبت تمامی المان های موجود در پنجره را دارد
- Restart : تمامی اطلاعات نقشه را بازنویسی میکند
- Accident : یک شکل دریافت کرده و وضعیت برخورد آن شکل را با دیوار یا خوراکی ها مشخص میکند
- setRandomFood : این متد با دریافت نوع خوراکی یکی از خوراکی های صفحه را تبدیل به خوراکی مورد نظر میکند .
- checkSpecoalFood : این تابع که در حلقه اصلی بازی دایما فرا خوانده میشود وظیفه چک کردن خوراکی ویژه موجود در نقشه را دارد . درصورتی که به مقدار زمان مشخص شده از قبل گذشته باشد آن خوراکی ویژه را محو کرده و خوراکی عادی را جایگزین آن میکند .

1.3.2 Animation :

کلاس انیمیشن تا حدی جامع و کاربردی طراحی شده تا در هر کلاس دیگری از پروژه بتوان از آن استفاده کرد.

```
private:
   sf::Shape * shape;
   sf::Clock clock;
   sf::Time time;
   int frame = 0;
   std::vector<sf::Texture> frames;
   bool enable = false;
public:
   Animation(sf::Shape * sh, std::string s, float second);
   void addTexture(std::string s);
   void update();
   void setenable(bool en);
```

- متد سازنده این کلاس ، یک اشاره گر به شکلِ از قبل ساخته شده ، یک استرینگ که نام فایل تصویر پیشفرض انیمیشن را در خود دارد و همچنین یک عدد float که بیانگر زمان آپدیت هر فریم به ثانیه میباشد را دریافت میکند .
- متغیر frame شماره فریمی که در حال حاضر در حال نمایش دادن به کاربر میباشد را ذخیره میکند .
- متغیر enable دو مقدار فعال و غیر فعال را برای انیمیشن در نظر میگیرد که در حالت غیر فعال انیمیشن ثابت شده و بر روی یک فریم باقی میماند .
- متد addTexture یک فریم را دریافت و به لیست فریم ها اضافه میکند .
- متد update که در حلقه اصلی بازی باید فرا خوانده شود برسی میکند که اگر زمان مورد نظر فرا رسیده باشد فریم انیمیشن را به بعدی انتقال دهد . (در صورت غیر فعال بودن انیمیشن این عمل صورت نمیگیرد)

1.3.3 Food :

```
enum class Type {Normal, Strength, Apple};
```

کلاس food داخل فضای نام fd میباشد ، یک enum class به اسم type دارد که انواع موجود یک غذا را مشخص میکند .

```
class Food : public sf::CircleShape
{
private:
   Type type;
   bool visible = true;
   sf::Texture txtapple;
public:
   Food();

   void change(Type t);
   Type getType();
   void setVisibility(bool v);
   bool getVisibility();
};
```

این کلاس که یک کلاس پایه است برای مدیریت خوراکی های بازی در کلاس نقشه استفاده میشود .

کلاس خوراکی ارث برده شده از کلاس (شکل دایره) میباشد تا در زمان ساخت ، یک دایره با ویژگی های مورد نظر ایجاد شده و ویژگی های یک خوراکی را نیز دارا باشد .

1.3.4 Menu :

کلاس منو شامل لیبل ها و کلید هایی میباشد که رابط کاربری بسیار ساده ای را برای توسعه بازی میدهد .

برای ساخت منو با استفاده از این کلاس فقط کافیست از دو متد addLabel و addButton المان های مورد نیاز خود را به منو اضافه کرده و متد draw را برای رسم منو در صفحه فراخوانی کنید .

نمونه ای از طراحی منو :

```
Menu menu(window, 400, 200, sf::Color::White, sf::Color::Red);
menu.addLabel("you lost!", 20, -1, 10, sf::Color::Red)
    .addLabel("your score : " + std::to_string(score), 20, -1, 40, sf::Color::Black)
    .addLabel("highest score : " + std::to_string(memory::getHighestScore()), 20, -1, 70, sf::Color::Black)
    .addButton("play again", 30, 70, 130, sf::Color::Black)
    .addButton("exit", 30, 270, 130, sf::Color::Black)
    .open();
```

لازم به ذکر است برای هر لیبل یا کلید جدید میتوان لوکیشن x و y را تعیین کرد ولی با وارد کردن مقدار منفی یک (1-) در این بخش آن لیبل یا کلید وسط چین میشود .

1.3.5 Move :

```
Move(Map & m, sf::RectangleShape & sh, float t);

void step();
void back();
void setRotate(DIRECTION dir);
void setenable(bool en);
```

کلاس move در ابتدا دو اشاره گر به یک نقشه و یک شکل را دریافت میکند و وظیفه حرکت دادن شکل در نقشه را دارد . (این کلاس برای حرکت پکمن ساخته شده ولی میتواند در جاهای دیگر هم مورد استفاده قرار گیرد)

با هر بار فراخوانی متد step در صورت فرا رسیدن زمان مناسب ، شکل به اندازه ی یک قدم در نقشه حرکت میکند .

متد setRotate (متد عمومی) جهت حرکت جدیدی را برای شکل ثبت میکند . متد rotate (متد خصوصی) که در هر قدم صدا زده میشود دایما در حال چک کردن جهت حرکت جدید میباشد ، در صورت امکان (برخورد نکردن به دیوار) جهت حرکت شکل را به جهت جدید تغییر میدهد .

متد back نیز با استفاده از lastMove آخرین حرکت شکل را بر میگرداند . (در زمان برخورد شکل با دیوار های نقشه این اتفاق صورت میگیرد)

1.3.6 Pacman :

```
Map & map;
sf::RectangleShape pacman;
Animation animation;
Move move;
int hp = 3;
int score = 0;
```

پکمن که تعریفی برای پلیر بازی میباشد از یک shape ، move ، animation و همچنین یک اشاره گر به نقشه ای که پلیر در آن قرار دارد ساخته شده است . علاوه بر این ویژگی ها کلاس پکمن شامل مقدار score و hp پلیر هم میباشد .

تمام فعالیت های پکمن در کلاس های پایه قبلی تعریف شده که باعث ساده شدن ساختار این کلاس میشود .

```
void Pacman::rotate(DIRECTION dir)
{
   move.setRotate(dir);
}
```

برای مثال برای چرخش پکمن از متد چرخش که در کلاس move تعریف شده استفاده میشود .

```
void Pacman::update()
{
   animation.update();
   move.step();
}
```

و یا برای بروز کردن پکمن در هر بار اجرای حلقه اصلی کافیست انیمیشن و حرکت پکمن را بروز کنیم .

1.3.7 RandomizeMove :

```
   RandomizeMove(Map & map, sf::RectangleShape & shape, float time);
   _DIRECTION step();
```

این کلاس که برای حرکت رندوم روح ها ساخته شده مانند کلاس move دو اشاره گر بـه یـک شـکل و یک نقشه را دریافت میکند و وظیفه حرکت رندوم شکل در نقشه را دارا میباشد .

این کلاس فقط یک تابع عمومی دارد که در حلقه اصلی بازی فراخوانی میشـود . در صـورت لـزوم خـود کلاس تصمیم به حرکت و یا چرخش میگیرد و شکل مورد نظر را در تصویر به حرکت وادار میکند .

1.3.8 Ghost :

```
Ghost(Map & map, float speed);
~Ghost();
void changeAnimationStatus(AnimationStatus animationStatus);

void update();
sf::RectangleShape getShape();
AnimationStatus getStatus();
```

در کلاس روح هم مانند پکمن تمام واکنش ها از پیش طراحی شده و به همین دلیل است کـه سـاختار این کلاس بسیار ساده میباشد .

مهم ترین متد این کلاس changeAnimationStatus میباشد که وظیفه تغییر وضعیت روح را دارد .

وضعیت های یک روح عبارت اند از : {Left, Right, DiedLeft, DiedRight, Scared} [2]

در صورتی که پکمن یک خوراکی قدرتی را تغذیه کند این تابع صـدا زده شـده و مقـدار نـوع (scared) برای روح ثبت میشود . مکانیزم برگشت حالت روح به صورت خودکار و با استفاده از کـلاس update رخ میدهد .

[2] لازم به ذکر است که این وضعیت ها اضافه شده ، ولی تکمیل نشده اند . البته پروژه به نحوی پیاده سازی شده است که میتوان در زمان کمی این وضعیت ها را تکمیل و یا وضعیت جدیدی برای روح ها تنظیم کرد .

1.3.9 Memory :

```
namespace memory
{
   void saveHighestScore(int s)
   {
      std::ofstream file("DB/the_highest_score.dat", std::ios_base::out |
std::ios_base::trunc | std::ios_base::binary);
      if (!file.is_open())
         throw 1;
      file.write((char*)&s, sizeof(int));
      file.close();
   }
   int getHighestScore()
   {
      int s;
      std::ifstream file("DB/the_highest_score.dat", std::ios_base::in |
 std::ios_base::binary);
      file.read((char*)&s, sizeof(int));
      file.close();
      return s;
   }
} // namespace memory
```

فضای نام memory دو تابع را دارا میباشد که یکی وظیفه ذخیره و دیگری وظیفه خواندن بیشترین امتیاز در دیتابیس پروژه را دارند .

2 گرافیک :

برای رابط گرافیکی این پروژه از کتابخانه قدرتمند sfml کمک گرفته شده است که توانایی های زیـادی ازجمله ساخت و ویرایش اشکال (مستطیل ، دایره ، چند ضلعی هـای منـتظم و غیـر منـتظم) ، تنظـیم دوربین و حرکت آن در زمینه برنامه ، ثبت و ویرایش صدا ها و موسیقی بـرای بـازی یـا برنامـه ، اضـافه کردن انواع متون با فونت های استاندارد یا اختصاصی را دارا میباشد .

همچینین این کتابخانه اجازه برسی برخورد دو شکل با یکدیگر را میدهد که در بازی میتـوان از تصـادف دو شی با خبر شد و نتیجه مورد نظر را برای آن ایجاد کرد .

در صورت استفاده از این کتابخانه برای نمایش اشیا در صفحه میتوان از اشکال یا اسـپرایت هـا اسـتفاده کرد . اسپرایت ها در مقابل اشکال از سرعت بالا تری برخوردارند و مختص رندر عکس ها طراحـی شـده اند ولی اشکال از امکانات بیشتری برخوردار میباشند که بسته به نیاز ، توسعه دهنده تصمیم به اسـتفاده یکی از این دو مورد میگیرد .

در این پروژه به دلیل نیاز به برسی برخورد المان های بازی از اشکال استفاده شده است .

3 توضیحات طراحی شی گرا

طبق قواعد برنامه نویسی شی گرا ، در این پروژه بـرای هـر یـک از تعـاریف (پکمـن ، روح ، نقشـه ، ...) کلاسی تعریف شده که شامل ویژگی ها و فعالیت هاییست که میتوانند انجام دهند .

پس از تعاریف اولیه و ساختن کلاس ها نوبت به تکمیل پروژه میشود . در این مرحله کافیست از کلاس های ساخته شده یک یا چند شی بسازیم و با متد های از پیش تعیین شده آنها را مدیریت کرده و بـازی را پیش ببریم .

البته در بعضی از مواقع (برای مثال ثبت کردن بیشترین رکورد) نیازی به نوشتن کـلاس و تعریـف اشـیا برای هربار استفاده نیست و یک تابع میتواند کار مورد نظر را انجام دهد . در این مورد برای دسته بنـدی توابع و موضوعیت بخشی به آنها نیاز است که از فضای نام استفاده کرده و توابع خـود را دسـته بنـدی و استفاده آنها را ساده تر کنیم . برای مثال در پروژه از فضای نام (memory) برای دو تابع ثبت و خواندن بیشترین رکورد در حافظه دایمی استفاده شده است .

توضیحات بیشتر و کامل تر درباره کلاس های پایه پروژه را میتوانید در بخش **کلاس های پایه** مشاهده کنید .

4 چالش ها

یکی از اصلی ترین دلایل ایجاد پروژه در هنگام آموزش (به خصوص آموزش برنامه نویسی) برخورد با چالش های مختلف و رفع آن چالش ها میباشد . در این راه است که بسیاری از مساله ها در ذهن برنامه نویس حل شده و باعث آشنایی هرچه بیشتر توسعه دهنده با تکنولوژی های مختلف و بروز میشود .

چالش اصلی بازی پکمن رابط گرافیکی آن بود که توسعه دهنده را از دنیای خط فرمان (command line) وارد دنیای جدیدی میکند . دنیایی پر از رنگ و لعاب که اجازه ارتباط هرچه بهتر با کاربر را محیا میکند . هرچند که خط فرمان در بین برنامه نویس ها راه ارتباطی محبوب تری است ولی در دنیای امروز کاربران انتظارات بیشتری از کامپیوتر دارند .

البته ، قابل ذکر است ، که sfml کتابخانه جامع و کاملی است که چالش های طراحی گرافیکی را بسیار کم کرده و تنها با چند خط کد و استفاده از متد های از پیش ساخته شده میتوان صفحه گرافیکی را ایجاد و المان های مختلفی را در آن ایجاد یا ویرایش کرد . ولی با این حال مدیریت فرایند (1.چک کردن رویداد(event) های ورودی 2.ایجاد تغییرات 3.ثبت تغییرات) که برای کار با این کتابخانه توصیه شده برای اولین بار کار به نسبت چالش بر انگیزی بود .

4.1 چالش زمان :

در بین تمام چالش های کار با این کتابخانه ، مدیریت زمان و برسی سرعت حرکت و انیمیشن بیشترین دغدغه و مشکل را ایجاد کرد . بازی باید هریک از دستورالعمل های تکرار شونده را با سرعت پردازنده و یا در یک مدت زمان نسبتا ثابت انجام دهد که کتابخانه sfml برای مدیریت این چالش دو کلاس sf::clock و sf::time را از پیش ساخته و در اختیار توسعه دهنده قرار داده است .

و اما حل این مشکل در پروژه پکمن :

```
void Move::step()
{
   if (clock.getElapsedTime().asSeconds() > time.asSeconds() && enable)
   {
      clock.restart();
      rotate();
      shape.move(lastMove);
   }
}
```

برای مثال در کلاس **move** متد step در یک حلقه بر اساس سرعت پردازنده ده ها و یا صد هـا بـار در ثانیه اجرا میشود . با دستور شرطی نوشته شـده برسـی میکنـیم کـه در صـورت رسـیدن زمـان موعـود دستورات لازم اجرا شود ، در غیر این صورت به انتهای متد رسیده و دیگر کـد هـای مـورد نیـاز را اجـرا میکند .

4.2 چالش برخورد با دیوار :

یکی از چالش های ساخت این بازی برسی برخورد پکمن و روح ها به دیوار بود .

```
void RandomizeMove::back()
{
   shape.move(-lastMove);
}
```

برای حل این مشکل در کلاس move و randomizeMove که بـه ترتیـب بـرای حرکـت pacman و ghost ایجاد شدند یک وکتور وجود دارد که آخرین حرکت شی را ثبت و در خود ذخیره میکند .

پس از برسی ها ، اگر شی به هریک از بلاک های دیـوار برخـورد داشــته باشـد متـد ()back فراخـوانی میشود .

این ترفند مانند این میباشد که پکمن یک قدم رو به جلو بر میدارد ، درصورت برخورد بـه دیـوار همـان قدم را بلعکس برداشته و به عقب بر میگردد . در نتیجه مانند این است که هیچ حرکتی نکرده است .

5 برامد های آموزشی

5.1 شی گرایی

ساخت پروژه در هر سطحی برای هر برنامه نویسی آموزش های بسیار زیاد و متنوعی را در پی دارد . ولی عامل اصلی برتری این پروژه نسبت به دیگر پروژه ها ساخت یک بازی میباشد . ساخت بازی برای درک هرچه بهتر مفهوم شی گرایی (کلاس ها ، اشیا ، متد ها ، ویژگی ها ، ...) برای برنامه نویسان توصیه میشود .

برای مثال نوشتن کلاسی که ویژگی های یک روح را توصیف میکند و همچنین کار هایی که یک روح میتواند انجام دهد . سپس ساختن اشیا از این کلاس و استفاده از آنها میتواند مفهوم شی گرایی را بهترین نحو شرح دهد .

5.2 چند ریختی

در برنامه نویسی شی گرا ، چند ریختی مساله ای بود که به شخصه تا قبل این پروژه به آن اهمیت زیادی نمیدادم . ولی در این پروژه در بخش انیمیشن مربوط به روح ها به دلیل استفاده بیرویه از دستور switch مجبور به استفاده از این ویژگی شدم . پس از استفاده از چند ریختی در پروژه حال با اضافه کردن چند خط میتوان انیمیشن های دیگری نیز برای حالات مختلف روح اضافه کرد .

5.3 رابط گرافیکی

کار کردن با کتابخانه sfml تجربه جالبی بود که در بخش چالش ها ، چالش های کار بـا ایـن کتابخانـه ذکر شده است .

6 منابع و ابزار

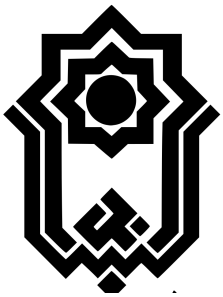
منابع استفاده شده برای برنامه نویسی سی پلاس پلاس

- Cppreference
- w3schools
- geeksforgeeks

منابع استفاده شده برای راه اندازی و استفاده از sfml

- YouTube
- sfml-dev.org

در این پروژه همچنین برای حل مشکلات و پاسخ به سوالات از بهترین دوست برنامه نویس ها stackoverflow استفاده شده .



PAC-MAN

Designed by

Hossein Haghighat

S.N

40012358010

1401-summer

Advanced Programming - Final Project

Dr. Morteza Yousef Saanaty