به نام خدا

گزارش کار

PACMAN

استاد:
دکتر مرتضی یوسف صنعتی

درس:
برنامه سازی پیشرفته

طراح:
علی میرزائی

دانشگاه بوعلی سینا

تابستان 1401

فهرست

مقدمه

بازی زیبای پکمن شاید از نظر گرافیکی پتانسیل لازم برای رقابت با بازی های مدرن را نداشته باشد، اما گیم پلی جذاب و خاطره انگیزی دارد.
در بازی پکمن شما محیطی دو بعدی و کاملا کارتونی را مشاهده خواهید کرد. کاراکتری که شما آن را کنترل می‌کنید یک توپ زرد رنگ است که دهانش باز و بسته می‌شود و مشخص است که منتظر خوردن اشیاء و متعلقات رو به رویش خواهد بود.
در مراحل اصلی بازی شما در مسیری پر پیچ و خم و شبیه به یک هزارراه قرار داده می‌شوید. وظیفه شما این است که با هدایت دقیق دایره زرد رنگ بازی تمام خوراکی ها و نقاطی که در مسیر های مختلف مپ وجود دارند را بخورید، اما در حین انجام این کار باید مراقب دشمنان شبیه به هشت پایی که در مسیر قرار دارند باشید، چرا که پکمن نمی‌توانید هشت پاها را بخورد و با برخورد به آن ها شما بازنده خواهید شد.
در ادامه نحوه پیاده سازی بازی پکمن را بررسی میکنیم.

نقشه بازی

طرح اولیه

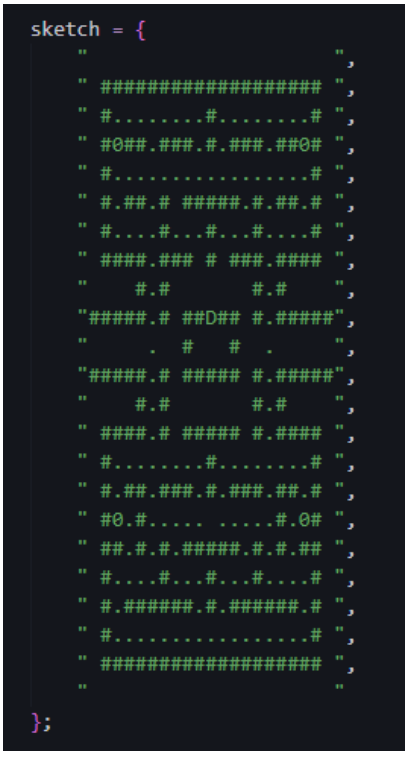
مراحل اماده سازی نقشه بازی در متد initMap از کلاس مپ انجام میشود و این متد هنگام ایجاد شی از کلاس مپ در کانستراکتور آن فراخوانی میشود.
برای رسم طرح اولیه بازی یک ارایه رشته ای به ارتفاع صفحه بازی درنظر گرفتم تا با کرکترها چیدمان نقشه بازی را پیاده سازی کنم.

```
std::array<std::string, Height> sketch;
```

در طرح اولیه، جزئیات صفحه بازی مانند

- دیوارها (#)
- خوراکی ها (.)
- خوراکی های قدرتی (0)
- درِ خانه ارواح (D)

همچنین فضاهای خالی در نظر گرفته شده است. طول رشته های هر سطر به اندازه طول صفحه بازی است.



تبدیل طرح اولیه به نقشه

برای تبدیل طرح اولیه به نقشه و نمایش آن از sfml کمک میگیریم. به این منظور از

- مربع برای دیوارها
- دایره با سایز کوچک برای خوراکی ها
- دایره با سایز بزرگ برای خوراکی های قدرتی

استفاده میکنیم. سایز های استفاده شده برای ساخت اشیا در فایل info.hpp ذخیره شده است.

در ذخیره سازی خانه ها برای دیوارها از ارایه و برای خوراکی های عادی و قدرتی از دو وکتور جداگانه استفاده میشود.

حال برای تبدیل کرکتر ها به اشکال تعریف شده، روی خانه های طرح اولیه سوییچ میکنیم و کرکتر کرکتر جلو میرویم. هرجا به # رسیدیم رنگ دیوار(مربع در نظر گرفته شده برای دیوار) را آبی میکنیم، هرجا به خانه های خالی رسیدیم آن را سیاه و آن را در ارایه مربوط به نقشه (map) ذخیره میکنیم. حالا یک ارایه با دیوارها و فضاهای خالی داریم.

برای خانه های دیگر هم به همین روش کار میکنیم و در انتها همه کرکترهای ما به خانه های نقشه تبدیل شده اند. برای رسم آن ها با گرفتن پنجره بازی به صورت رفرنس، آن ها را با استفاده از متد draw رسم میکنیم.

```
switch (sketch[i][j])
{
case '#':
    Wall.setFillColor(Color::Blue);
    Wall.setPosition(Vector2f(j * CellSize, i * CellSize));
    map[i][j] = Wall;
    break;
case ' ':
    Wall.setFillColor(Color::Black);
    Wall.setPosition(Vector2f(j * CellSize, i * CellSize));
    map[i][j] = Wall;
    break;
case '.':
    food.setPosition(Vector2f((j * CellSize) + 10, (i * CellSize) + 10));
    foods.push_back(food);
    break;
case '0':
    power.setPosition(Vector2f((j * CellSize) + 5, (i * CellSize) + 5));
    powerFood.push_back(power);
    break;
case 'D':
    RectangleShape door(Vector2f(CellSize, CellSize / 2));
    door.setFillColor(Color::White);
    door.setPosition(Vector2f(j * CellSize, i * CellSize));
    map[i][j] = door;
    break;
}
```

متد checkEndLevel

در این متد بررسی میشود اگر هیچ خوراکی ای روی نقشه باقی نمانده باشد، نقشه دوباره چیده میشود و به مرحله بعد میرود. همچنین این متد خروجی bool دارد و درصورت رفتن به مرحله مقدار true برمیگرداند.

خوراکی های ویژه

خوراکی های ویژه یا همان میوه ها در مراحل بازی به شکل های مختلف و به مدت ده ثانیه ظاهر میشوند. برای هرکدام از میوه ها تکسچر در نظر گرفته شده که تصاویر آن ها در این تکسچرها لود میشوند.
تصاویر مربوط به میوه ها از دایرکتوری پروژه در مسیر "Photo/fruit/" قرار گرفته اند.
همچنین یک اسپرایت Fruit تعریف شده که در هر مرحله با توجه به میوه آن مرحله، تکسچر مربوطه را در اسپرایت ست میکند.

متد createFruit

این متد زمانی که خوراکی های خورده شده توسط پکمن به حد قابل قبولی (70 و 135 عدد خوراکی) برسد فرخوانی میشود.
دو عدد رندوم به عنوان x و y ذخیره میکند. برای اینکه این دو عدد قابل قبول باشند و میوه بتواند در آن مکان ظاهر شود باید دو شرط رعایت شده باشد:

- با موقعیت فعلی پکمن برخورد نداشته باشد
- روی خوراکی های موجود در نقشه ظاهر نشود

در غیر اینصورت دوباره دو عدد رندوم دیگر تولید میشود و همین مراحل تکرار میشود تا زمانی که دو عدد شروط لازم را دارا باشند. بعد از آن اسپرایت میوه مورد نظر با موقعیت مناسب در وکتور fruit ذخیره میشود.

متد update

در این متد بررسی میشود که اگر میوه ای 10 ثانیه در صفحه بازی بوده و توسط پکمن خورده نشده، از صفحه بازی پاک شود.
همچنین متد checkEndLevel را فراخوانی میکند.

پکمن

ساخت

یک تکسچر به شکل دایره برای پکمن درنظر گرفته شده که از دایرکتوری پروژه در مسیر "Photo/pacman/" قرار گرفته است. همچنین برای نمایش پکمن از یک اسپرایت به نام player استفاده میشود که در متد initVariables مقداردهی اولیه میشود.

متد Move

این متد برای کنترل و حرکت دادن پکمن استفاده میشود. برای اینکار یک متغیر از نوع عدد صحیح به نام dir درنظر گرفته شده که با مقادیر:

- 0 : حرکت به راست
- 1 : حرکت به پایین
- 2 : حرکت به چپ
- 3 : حرکت به بالا

مقداردهی میشود.
زمانی که کلیدی از کلیدهای حرکت به 4 طرف توسط کاربر فشرده میشود، جهت حرکت آن کلید در dir ذخیره میشود و به کمک دستور سوییچ با توجه به dir موقعیت پکمن تغییر میکند. بدین ترتیب زمانی که کاربر کلیدی را انتخاب نمیکند، اخرین کلید انتخاب شده در dir باقی می ماند و پکمن به همان سمت حرکت میکند.

```
switch (dir)
{
case 0:
    position.x += pacmanSpeed;
    break;
case 1:
    position.y += pacmanSpeed;
    break;
case 2:
    position.x -= pacmanSpeed;
    break;
case 3:
    position.y -= pacmanSpeed;
    break;
}
```

برخورد با دیوارها

پکمن نباید بتواند از دیوارها عبور کند. پس موقع حرکت باید دقت کنیم هنگامی که به دیوارها رسید متوقف شود. بدین منظور یک ارایه چهارتایی bool (wall) تعریف شده که وجود دیوار در چهار طرف پکمن را بررسی میکند. اگر دیوار وجود داشته باشد با 1 و اگر نباشد با 0 مقداردهی میشود.
موقعیت بعدی پکمن در هرجهت با توجه به سرعت آن را به تابع collision (نحوه کارکرد این تابع را در ادامه بررسی میشود) میدهیم تا این تابع بررسی کند پکمن میتواند در این مسیر حرکت کند یا نه(به دیوار برخورد نکند) و آن را در wall ذخیره میکند.

```
// Collision Right
wall[0] = collision(player.getPosition().x + Speed - 15, player.getPosition().y - 15, map);
// Collision Down
wall[1] = collision(player.getPosition().x - 15, player.getPosition().y + Speed - 15, map);
// Collision Left
wall[2] = collision(player.getPosition().x - Speed - 15, player.getPosition().y - 15, map);
// Collision Up
wall[3] = collision(player.getPosition().x - 15, player.getPosition().y - Speed - 15, map);
```

هنگامی که کاربر کلیدی رو میزند و میخواهد در آن جهت حرکت کند بررسی میکنیم آیا پکمن مجاز به حرکت در آن مسیر هست یا نه. اگر مجاز باشد dir دلخواه کاربر ست میشود و پکمن در آن جهت حرکت میکند و درغیر اینصورت به حرکت در مسیر قبلی خود ادامه میدهد تا به دیوار برسید و متوقف شود.

تابع Collision

این تابع موقعیتی را دریافت میکند و یک مربع به اندازه پکمن و در همان موقعیت میسازد. حال روی خانه های نقشه که به این تابع پاس شده حرکت میکنیم.
هرجای نقشه که مربع ما با آن برخورد کند و رنگ نقشه در آن نقطه آبی باشد(فضاهای خالی با مربع های سیاه مقداردهی شده اند) تابع مقدار 1 را برمیگرداند(در این موقعیت دیوار وجود دارد). اگر تمام خانه های نقشه را بررسی کنیم و این اتفاق رخ ندهد مقدار 0 برا برمیگرداند(برای حرکت در این جهت مشکلی وجود ندارد).

```
if (wall.getGlobalBounds().intersects(map[j][i].getGlobalBounds())
&& map[j][i].getFillColor() == Color::Blue)
{
    return 1;
}
```

کنترل بهتر

در کنترل پکمن با مشکلی مواجهیم. هنگامی که بخواهیم به سمتی حرکت کنیم، باید دقیقا زمانی که مسیر باز است و دیوار در آن وجود ندارد کلید مربوطه را بزنیم یا قبل از آن کلید را فشرده نگه داریم تا بتوانیم به سمت دلخواه حرکت کنیم.
برای حل این مشکل از متغیری از نوع عدد صحیح (nextDir) که مقادیری همانند dir میگیرد استفاده میکنیم. درحالت قبل اگر کاربر قصد حرکت به سمتی را داشت و در آن لحظه امکان حرکت در آن مسیر وجود نداشت، پکمن به حرکت در مسیر خود ادامه میداد. در اینجا میتوانیم از متغیر کمکی (nextDir) استفاده کنیم.
در حالتی که کاربر میخواهد به جهتی برود و امکان آن وجود ندارد، مسیر انتخاب شده توسط کاربر در nextDir ذخیره میشود و در اولین جایی که ممکن باشد در مسیر دلخواه کاربر قرار میگیرد.
در این صورت دیگر نیازی به نگه داشتن دکمه یا عکس العمل خاصی برای حرکت در مسیر دلخواه وجود ندارد.

انیمیشن

برای انیمیت کردن پکمن از متد های animation و Rotate استفاده میشود.

متد animation

سه تکسچر برای پکمن درنظر گرفته شده که در مسیر "Photo/pacman/" قرار گرفته اند. در متد animation از یک متغیر به نام frame برای جا به جایی بین این 3 حالت استفاده میشود.
هر 0.08 ثانیه یکبار بین مقادیر مختلف frame (از 1 تا 4) سوییچ میکنیم.
تکسچر اول را ست میکنیم و به مقدار frame یک واحد اضافه میکنیم. 0.08 ثانیه بعد تکسچر دوم را ست میکنیم و به همین ترتیب ادامه میدهیم تا frame برابر با 4 شود.
در اینجا آن را برابر با 1 قرار میدهیم تا این چرخه ادامه پیدا کند.
همچنین برای فراخوانی این توابع دو شرط تعریف شده است:

- کاربر بازی را شروع کرده باشد
- پکمن درحال حرکت باشد

چرا که وقتی پکمن به دیوار برخورد میکند و متوقف میشود، انیمیشن نباید فعال باشد.

متد Rotate

حال پکمن با انیمیشن حرکت میکند اما هنگام تغییر مسیر به طرفین جهت صورت آن تغییر نمیکند و همیشه به سمت راست است. در اینجا از متد Rotate کمک میگیریم.

چرخاندن پکمن در دو مرحله انجام میشود. در روند اجرای برنامه اخرین جهتی که پکمن درحال حرکت در آن بوده در متغیر previousDir ذخیره میشود.

در مرحله اول با توجه به اخرین جهت حرکت پکمن آن را به حالت اولیه (سمت راست) برمیگردانیم.

```
switch (previousDir)
{
case 0://Right
    break;
case 1://Down
    player.rotate(-90);
    break;
case 2://Left
    player.rotate(-180);
    break;
case 3: //Up
    player.rotate(-270);
    break;
}
```

در مرحله دوم با توجه به جهت فعلی پکمن و با استفاده از متد rotate که sfml در اختیار ما گذاشته پکمن را میچرخانیم.

```
switch (dir)
{
case 0://Right
    break;
case 1://Down
    player.rotate(90);
    break;
case 2://Left
    player.rotate(180);
    break;
case 3://Up
    player.rotate(270);
    break;
}
```

جزئیات

استفاده از تونل

در دو طرف نقشه بازی دو تونل وجود دارد که پکمن میتواند از یکی خارج و از دیگری وارد شود. هنگامی که مولفه x موقعیت پکمن بیشتر یا کمتر از صفحه بازی باشد میفهمیم پکمن وارد تونل ها شده است. با توجه به تونلی که پکمن وارد آن شده موقعیت آن را تغییر میدهیم تا از تونل دیگر وارد شود.

```
//Use tunnel
if (player.getPosition().x - 15 >= CellSize * Width)//Exit from right side of the map
{
    position.x =  15 - CellSize;
    player.setPosition(Vector2f(position.x , player.getPosition().y));

} else if (player.getPosition().x - 15 <= -CellSize)//Exit from left side of the map
{
    position.x = Width * CellSize + 15;
    player.setPosition(Vector2f(position.x, player.getPosition().y));
}
```

متد Reset

بعد از اینکه پکمن با روح ها برخورد کرد و جان از دست داد، به موقعیت اولیه برگردد.

متد resetPacman

بعد از اینکه پکمن 3 جان خود را از دست داد و بازی تمام شد، برای شروع دوباره بازی، این متد تمام متغیرها را به حالت اولیه برمیگرداند.

متد eat

وقتی پکمن با خوراکی ها برخورد میکند، براساس خوراکی خورده شده به امتیاز کاربر اضافه و خوراکی خورده شده از صفحه بازی حذف شود.

متد accident

اسپرایت روح ها را میگیرد و بررسی میکند آیا با پکمن برخورد داشته اند یا نه.

متد createFruit

در هر مرحله هنگامی که پکمن تعداد 70 یا 135 خوراکی بخورد به مپ اطلاع میدهد تا خوراکی ویژه (میوه) تولید کند.

روح ها

ساخت

در زمین بازی، 4 روح با رنگ های مختلف وجود دارد که برای هرکدام تکسچر مناسب با آن را در مسیر "Photo/ghost/" قرار گرفته و در کانستراکتور کلاس Ghost لود میشوند. همچنین برای هر روح یک اسپرایت در نظر گرفته شده که تکسچر مناسب با آن در متد initGhosts ست میشود.

حرکت

مانند پکمن برای هرکدام از روح ها یک متغیر (ghostDir) برای حرکت درنظر گرفته شده که جهت حرکت روح را تعیین و با استفاده از دستور سوییچ، روح را به حرکت درمی آورد.

```
switch (ghostDir[i])
{
case 0:
    position.x += Speed;
    break;
case 1:
    position.y += Speed;
    break;
case 2:
    position.x -= Speed;
    break;
case 3:
    position.y -= Speed;
    break;
}
```

برخورد با دیوارها

همه چیز مطابق آنچه است که برای پکمن توضیح داده شد. فقط یک تفاوت کوچک وجود دارد. درِ خانه روح ها که با رنگ سفید نمایش داده شده، همانند دیوارها عمل میکند و روح ها نمیتوانند از آن خارج شوند.

تابع Collision

تابع collision به عنوان پارامتر اخر یک متغیر door میگیرد که تعیین میکند کسی بتواند از در عبور کند یا نه. این پارامتر به صورت دیفالت با 0 مقداردهی شده و به معنی آن است که کسی اجازه عبور از آن را ندارد (1 برابر اجازه عبور از دیوار است).
حال میخواهیم وقتی روح ها درون خانه هستند اجازه خروج داشته باشند، اما از بیرون نتوانند وارد خانه شوند.
همانطور که در توضیحات تابع collision پکمن گفته شد، وجود دیوار در 4 طرف روح ها بررسی میشود. هنگامی که روح ها میخواهند از در خارج شوند به سمت بالا حرکت میکنند. پس زمانی که میخواهیم نقاط برخورد به سمت بالا را بررسی کنیم میتوانیم به عنوان پارامتر آخر door را با 1 مقداردهی کنیم تا برای عبور از در مشکلی نداشته باشیم. چون از جهات دیگر door با 0 مقداردهی شده، اجازه ورود به خانه داده نخواهد شد.

```
// Collision Right
wall[0] = collision(ghost[i].getPosition().x - 15 + Speed, ghost[i].getPosition().y - 15, map);
// Collision Down
wall[1] = collision(ghost[i].getPosition().x - 15, ghost[i].getPosition().y + Speed - 15, map);
// Collision Left
wall[2] = collision(ghost[i].getPosition().x - 15 - Speed, ghost[i].getPosition().y - 15, map);
// Collision Up
wall[3] = collision(ghost[i].getPosition().x - 15, ghost[i].getPosition().y - Speed - 15, map, 1);
```

این شرط نادیده گرفته شده و در، همچون فضاهای خالی درنظر گرفته میشود.

```
if (door == 0 && wall.getGlobalBounds().intersects(map[j][i].getGlobalBounds())
&& map[j][i].getFillColor() == Color::White)
{
    return 1;
}
```

متد scatterMode

این متد جهت حرکت روح ها را مشخص میکند. بین جهت های قابل انتخاب (1 تا 4) به صورت رندوم انتخاب میکند و دو شرط لازم را برای آن بررسی میکند:

- در آن جهت دیواری نباشد

```
!wall[random]
```

- روح ها چرخش 180 درجه نداشته باشند

```
(random != dir - 2 && random != dir + 2)
```

در غیر اینصورت مراحل دوباره تکرار میشوند تا جهت مناسب پیدا شود.

یک متغیر کمکی (try1) در نظر گرفته شده تا اگر بعد از بررسی 50 عدد تصادفی، عددی مناسب پیدا نشد، روح را متوقف کند و با خارج شدن از حلقه ،مشکلی در روند اجرای برنامه ایجاد نشود.

انیمیشن

متد animation

پای روح ها هنگام حرکت انیمیشن دارد. همچنین روح ها به هرجهتی حرکت کنند، چشم آن ها باید آن سمت را نگاه کند. برای اینکار تکسچر های مورد نیاز از قبل لود شده اند.

هر 0.05 ثانیه یکبار انیمیشن اپدیت میشود. برای کنترل حرکت پای روح ها از متغیر frame کمک میگیریم. برای اینکار در زمان اپدیت انیمیشن باید بین دو تکسچر جا به جا شویم تا انیمیشن پای روح ها نمایش داده شود.

برای اینکه چشم روح ها مسیر حرکت را دنبال کند کافیست با توجه به جهت حرکت آن، تکسچر مربوطه را ست کنیم. برای اینکار کافیست تکسچرها را مرتب لود کرده باشیم.

پارامتر های داده شده از چپ:

i : رنگ روح را تعیین میکند

dir : جهت حرکت روح را تعیین میکند

frame : وضعیت پای روح را تعیین میکند

```
if (clock[i].getElapsedTime().asSeconds() > 0.05)
{
    clock[i].restart();
    ghost[i].setTexture(ghostTexture[i][dir][frame[i]]);

    if (frame[i] == 0)
        frame[i] = 1;
    else
        frame[i] = 0;
}
```

حالت ترسیده

هنگامی که پکمن خوراکی های قدرتی را میخورد، روح ها به مدت 5 ثانیه به حالت ترسیده میروند و شکل ظاهری آن ها تغییر میکند. در این حالت از یک متغیر زمانی به نام scaredTime استفاده شده که ریست میشود و تا 5 ثانیه تکسچرها را تغییر میدهد تکسچرهای حالت ترسیده در ارایه ای به نام scaredGhostTexture ذخیره شده اند.

2 ثانیه اول حالت ترسیده با حالت عادی تفاوت چندانی ندارد. فقط رنگ آن تغییر میکند و انیمیشن پاها همان است. حتی نیازی نیست جهت حرکت و چشم روح ها در یک راستا باشند.

```
if (clock[i].getElapsedTime().asSeconds() > 0.05)
{
    clock[i].restart();
    ghost[i].setTexture(scaredGhostTexture[wink[i]][frame[i]]);

    if (frame[i] == 0)
        frame[i] = 1;
    else
        frame[i] = 0;
}
```

در 3 ثانیه بعدی حالت ترسیده، روح ها باید چشمک بزنند و از رنگ آبی به سفید و برعکس تغییر رنگ بدهند.
برای جا به جایی بین تکسچرهای آبی و سفید از متغیری به نام wink استفاده میشود که با 0 و 1 مقداردهی میشود:

- 0 : تکسچرهای آبی رنگ
- 1 : تکسچرهای سفید رنگ

هر 0.09 ثانیه یک بار بین حالت سفید و آبی جا به جا میشوند و انیمیشن چشمک را به وجود می آورند.

```
if (scaredTimeWink[i].getElapsedTime().asSeconds() > 0.09)
{
    scaredTimeWink[i].restart();
    //The ghost flashes after 2 seconds of fear time
    if (scaredTime.getElapsedTime().asSeconds() >= 2)
    {
        if (wink[i] == 0)
            wink[i] = 1;
        else
            wink[i] = 0;
    }
}
```

منو بازی

صفحه آغازین بازی

متد gameMenu

هنگامی که بازی اجرا میشود، صفحه ای نمایش داده میشود که شامل 3 گزینه است:

PLAY : بازی کردن

SETTINGS : تنظیمات

QUIT : خروج از بازی

مدیریت ماوس

متد mouseHandling

این متد یک رشته به عنوان ورودی میگیرد و با توجه به این رشته متوجه میشود در کدام قسمت از منوی برنامه قرار داریم و با توجه به آن نکات مربوط به آن قسمت را اعمال میکند. همچنین به عنوان خروجی عدد صحیحی برمیگرداند که تعیین میکند آیا کاربر روی کلیدی کلیک کرده است یا نه. اگر کاربر گزینه ای را انتخاب نکرده باشد عدد 0 را برمیگرداند و اتفاق خاصی نمیفتد.

هنگامی که ماوس روی این نوشته ها قرار میگیرد، نوشته ها به رنگ زرد در می آیند تا کاربر متوجه شد این نوشته ها قابل کلیک کردن هستند.

```
if (play.getGlobalBounds().contains(mouse))
{
    play.setFillColor(Color::Yellow);
    if (Mouse::isButtonPressed(Mouse::Left))
    {
        return 1;
    }
} else
{
    play.setFillColor(Color::White);
}
```

تنظیمات

متد settingScreen

درصفحه اصلی منو، هنگامی که کاربر روی گزینه ستینگز کلیک میکند وارد صفحه ای میشود که دو گزینه در آن موجود است:

RESET HIGH SCORE : بیشترین امتیازی که گرفته اید، پاک میشود.

BACK TO MENU : به منوی اصلی بازی برمیگردد.

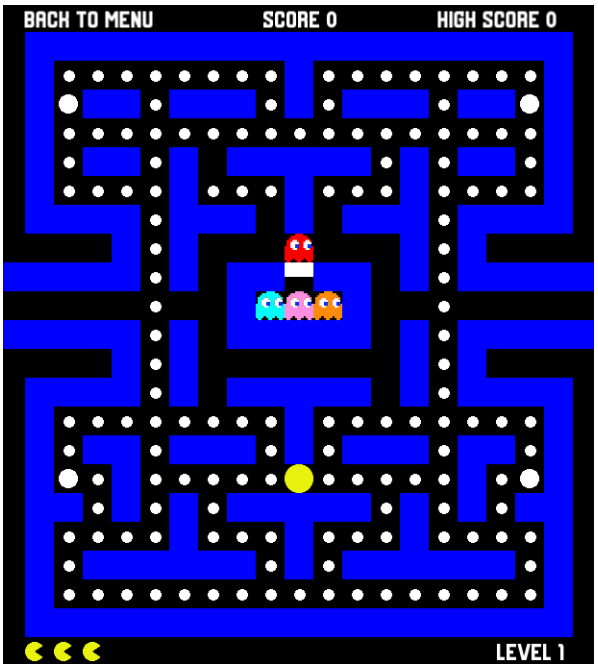
هنگامی که گزینه اول را انتخاب میکنید به مدت 1 ثانیه پیامی در پایین صفحه به شما نمایش داده میشود که متوجه شوید عملیات مورد نظر شما با موفقیت انجام شده است.

نمایش اطلاعات در صفحه بازی

متد draw

به کمک این متد در صفحه بازی اطلاعاتی به کاربر نمایش داده میشود:

- بالای صفحه سمت چپ : دکمه ای برای برگشت به منو
- بالای صفحه وسط : امتیاز لحظه ای
- بالای صفحه سمت راست : بیشترین امتیاز
- پایین صفحه سمت چپ : تعدادی پکمن به معنی تعداد جان ها
- پایین صفحه سمت راست : شماره مرحله



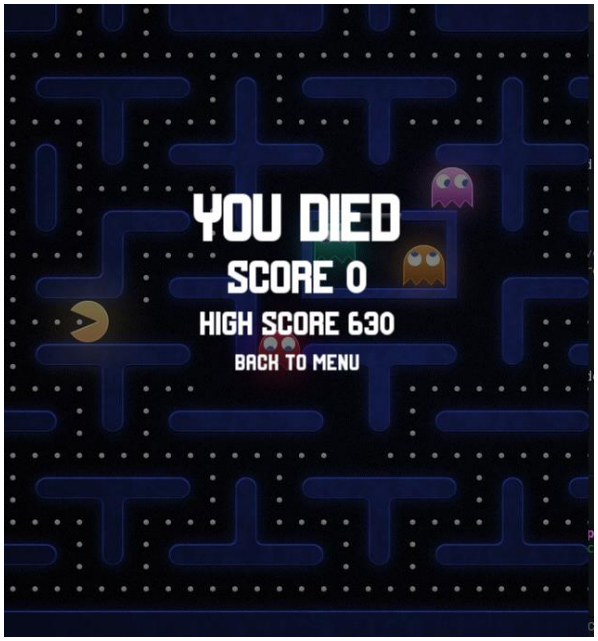
جزئیات

متد levelUpScreen

هنگامی که پکمن تمام خوراکی ها را میخورد، به مدت 1 ثانیه صفحه ای نمایش داده میشود که شماره مرحله جدید روی آن نوشته شده و بعد از آن مرحله جدید اغاز میشود.

متد deathScreen

هنگامی که کاربر تمام جان هایش را از دست میدهد میمیرد و اطلاعاتی به او نمایش داده میشود. از این صفحه میتواند وارد منوی اصلی بازی شود.



متد resetData

وقتی کاربر برروی دکمه RESET HIGH SCORE کلیک میکند، این متد فراخوانی میشود. در این متد ابتدا فایل مربوط به ذخیره بیشترین امتیاز باز میشود و بعد از پاک کردن اطلاعات موجود، عدد 0 در آن ذخیره میشود.

```
int score = 0;
fstream file;
file.open("../data/data.txt", ios::out);
file << score;
file.close();
```

منابع

منابعی که برای یادگیری sfml از آن ها استفاده شد:

کتاب : اموزش مقدماتی برنامه نویسی بازی با استفاده از sfml

دوره اموزشی : (332) [C++ & SFML - Simple 2D Games] - Introduction video - YouTube

دیگر منابع :

SFML community forums - Index (sfml-dev.org)

GeeksforGeeks | A computer science portal for geeks

Stack Overflow - Where Developers Learn, Share, & Build Careers

(332) Making the Original PACMAN in C++ and SFML - SFML Tutorial - YouTube

Google