|  |  |
| --- | --- |
| پروژه نهایی  بازی پک-من | **درس برنامه سازی پیشرفته**  **استاد درس**  **دکتر مرتضی یوسف صنعتی**  **دانشجو**  **عادل عسگری**  **40012358027**  **بهار 1401** |



# مقدمه

در این پروژه بر روی ساخت بازی قدیمی پک-من [[1]](#footnote-1) کار کرده ایم. بازی پک-من یک بازی ارکید محبوب و نوستالژی می باشد. این بازی محبوب جهانی ساخته ی استودیوی BANDAI NAMCO می باشد. و در ابتدا برای کنسول های قدیمی میکرو و سگا طراحی و ارایه شده بود. ولی به دلیل محبوبیت بالا حالا برای دستگاه های اندرویدی هم ارایه شده است. در این بازی بازیکن در مسیر های معینی حرکت می کند و باید مواظب دشمنان باشد. بازیکن در طول مسیر نیز به جمع اوری سکه و امتیاز می پردازد. زبان برنامه نویسی مورد استفاده در این پروژه c++ به همراه کتاب خانه SFML[[2]](#footnote-2) می باشد و در ابزار VSCODE نوشته شده است .

# تعریف کلی بازی

## مراحل

با توجه به کمبود وقت ، برای این بازی تنها یک مرحله در نظر گرفته شده است اما تمام قواعد بازی در این مرحله پیاده سازی شده است.

## حرکت

برای حرکت دادن پک-من از کلیدهای جهت دار یا کلیدهای (W-A-S-D)[[3]](#footnote-3) استفاده می کنیم. با زدن کلید های حرکتی پک من به حرکت در آمده و به اطراف می رود. اما باید در نظر گرفت که حرکت زمانی امکان پذیر است که به دیوار ها برخورد نکرده باشیم. دیوار ها مسیر بازی را ترسیم می کنند و از عبور روح ها و پک من از روی دیوار ها جلوگیری می کنیم.

## خوراک عادی

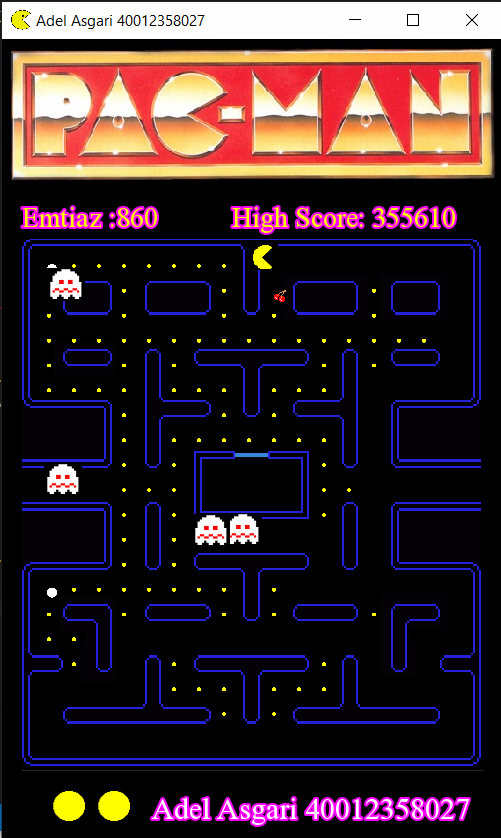
در محیط بازی یک سری نقطه وجود دارد که با خوردن[[4]](#footnote-4) هر کدام 10 امتیاز به مجموع امتیازات پک-من افزوده می شود. در صورتی که تمام این نقاط توسط پک-من خورده شود بازی با برد به پایان می رسد و می توان به مرحله بعدی رفت.

## خوراک قدرتی

در محیط بازی نقاط دایره ای توپر سفید رنگی قرار گرفته است که با خوردن آن ها 50 امتیاز به مجموع امتیازات افزوده می شود. بلافاصله پس از خوردن خوراک قدرتی روح ها به مدت 6 ثانیه[[5]](#footnote-5) در وضعیت ترسیده قرار گرفته و آبی رنگ می شوند و با چرخش 180 در جه ای سعی می کنند تا از مسیر قبلی دور شوند. قبل از اتمام 6 ثانیه 5 بار[[6]](#footnote-6) رنگ روح های ترسیده بین سفید و آبی تعویض می شود.(حالت چشمک زدن بین رنگ آبی و سفید)

## میوه

در مراحل مختلف بازی پس از خوردن تعدادی خوراک عادی میوه هایی به عنوان جایزه نمایش داده می شوند که 10 ثانیه بر روی تصویرو در مختصات تصادفی ظاهر می شوند و با خوردن هر کدام امتیاز مشخصی به پک-من تعلق می گیرد



شکل ‏2‑1 میوه گیلاس ظاهر شده در صفحه پس از خوردن 70 خوراک عادی

# معرفی بازی

این بازی مطابق با بازی قواعد تعریف شده در جزوه پروژه پیاده سازی شده است. در تصویر زیر محیط بازی را مشاهده می کنیم.



شکل ‏3‑1 محیط بازی پیاده سازی شده

بخش های شماره گذاری شده در بازی به شرح زیر می باشد:

1-بیشترین امتیاز کسب شده تا این لحظه در بازی می باشد. به محض اینکه این بازی به پایان برسد(برد یا باخت) بیشترین امتیاز به روز رسانی می شود.[[7]](#footnote-7)

2-امتیاز این بازیکن در این بازی تا این لحظه این امتیازا را کسب نموده است. امتیاز از طریق خوردن خوراکی های عادی ، خوراکی های قدرتی، خوردن روح ها و تغذیه از غذاهایی که به عنوان پاداش به ما تعلق می گیرد، کسب می شود.

3-روح های موجود در بازی 4 عدد می باشند که در این عکس در حال ترسیده قرار دارند.

4-پک-من یا بازیکن اصلی می باشد که با استفاده از کلیدهای جهت دار یا (W-A-S-D) در صفحه حرکت می کند.

5- تعداد جان های باقی مانده پک من در این بخش قابل مشاهده است. تعداد جان های پک-من 3 عدد می باشد که در ابتدای بازی یکی از جان ها در حال بازی می باشد و دو عدد دایره زرد رنگ در پایین صفحه نمایانگر تعداد جان های باقی مانده می باشد. با هر بار برخورد پک-من به روح هایی که در حالت تعقیب یا پرسه زدن باشد یکی از جان ها کم شده و 20 امتیاز نیز کسر می شود.

# گرافیک پروژه

## پنجره بازی

برای این بازی یک پنجره 800 \* 500 در نظر گرفته شده است. تمام ترسیمات بازی بر روی این پنجره انجام می شود. محدودیت حداکثر 60 فریم در ثانیه نیز برای این پنجره اعمال شده است.

RenderWindow MainWindow(VideoMode(500, 800), "Adel Asgari 40012358027");

## انیمشین بازی

برای ساخت انیمیشن های بازی از اشیایی از جنس spirit استفاده شده است. کافی است در فریم های مختلف برش هایی از یک عکس ثابت که حالت های مختلف یک کاراکتر را دارا می باشد را نمایش دهیم در نتیجه باعث می شود تا حرکت های یک کاراکتر مشابه یک انیمیشن دیده شود. [[8]](#footnote-8)

انیمیشن های کاراکترهای مختلف در وضعیت های متفاوتی اجرا می شوند. انیمیشن پک-من دائم تکرار می شود و با چرخش پک-من کاراکتر نیز می چرخد.



شکل ‏4‑1 اسپرایت مورد استفاده برای انیمیشن پک-من



شکل ‏4‑2 اسپرایت مورد استفاده برای باخت پک-من

انیمیشن مورد استفاده در روح ها نیز با کلیدهای جهت دار اجرا می شوند. زمانی که روح به هر سمتی می چرخد کاراکتر روح به شکلی تنظیم می شود که چشم های روح به آن سمت حرکت نماید. همچنین برای هر سمتی دو عکس در نظرگرفته شده است که حالت پا ها متفاوت می باشد که به تناوب نمایش داده می شود و این تصور ایجاد می شود که پاهای روح حرکت می کند.









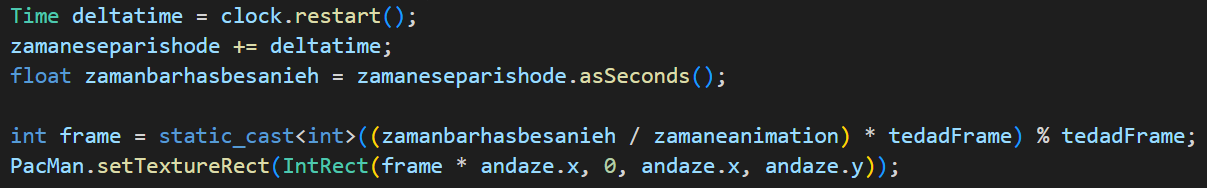
شکل ‏4‑3 اسپرایت های مورد استفاده برای روح های مختلف



شکل ‏4‑4 اسپرایت مورد استفاده برای روح های ترسیده



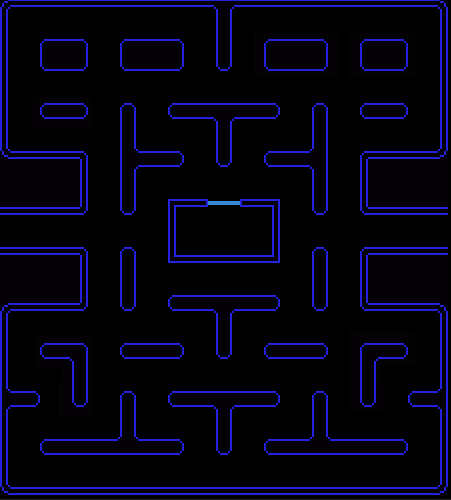
شکل ‏4‑5 اسپرایت مورد استفاده برای میوه ها

برای ایجاد انیمیشن پک-من کد زیر در حلقه اصلی بازی نوشته شده است. 

متغیر deltatime زمان هر بار اجرای حلقه بازی را در خورد ذخیره می کند. سپس کل زمان سپری شده بر حسب ثانیه را محاسبه می نماییم. فرض کرده ایم که زمان اجرای انیمیشن پک-من 0.2 ثانیه باشد و می دانیم انیمیشن پک من از 3 فریم تشکیل شده است. اکنون با تقسیم زمان سپری شده از بازی بر زمان انیمیشن (که 0.2 می باشد) تعداد انیمیشن های کامل به دست می آید و با ضرب در تعداد فریم (که 3 می باشد) ، تعداد کل فریم هایی که تا این لحظه پخش شده است محاسبه می شود. اکنون اگر باقی مانده این مقدار را بر تعداد فریم(که 3 می باشد) به دست بیاوریم متوجه می شویم که در این لحظه کدام فریم باید نمایش داده شود. سپس توسط تابع setTextureRect بخشی از شکل ‏4‑1 که مورد نیاز باید را برش داده و نمایش می دهیم.

## نقشه بازی

برای نقشه بازی از نگاشت یک ماتریس بر روی یک عکس استفاده شده است. عکس مورد استفاده در بخش زیر قابل مشاهده است.[[9]](#footnote-9)



شکل ‏4‑6 عکس مورد استفاده برای نقشه بازی

یک ماتریس 19 \* 22 (22 سطر و 19 ستون) به عنوان نقشه بازی در نظر گرفته شده است. سپس این ماتریس بر روی شکل ‏4‑5 نگاشت شده است تا گرافیک بازی شکل بگیرد. محتوای خانه ها با اعداد نمایش داده شده در جدول 1 مطابقت دارد.

|  |  |
| --- | --- |
| عدد | معنی |
| 0 | خوراک معمولی |
| 1 | خوراک قدرتی |
| 2 | دیوار |
| 3 | خالی (یعنی این خانه یا خوراکی بوده که خورده شده یا از ابتدا خالی بوده است ) |
| 5 | درِ خانه ارواح |

جدول ‏4‑1 مقادیر مورد استفاده در جدول نقشه

برای نگاشت ماتریس بر روی نقشه بازی از محاسبات ریاضی استفاده شده است. به عنوان مثال:



نقشه بازی از نقطه (200و20) ترسیم شده است. در نتیجه زمانی که بخواهیم نگاشت از مختصات جسمی بر روی ماتریس داشته باشیم باید از y آن نقطه 200 واحد و از x آن نقطه 20 واحد کم کنیم. همچنین در زمان ترسیم نقشه، برای نگاشت بهتر نقاط بر روی تصویر نقشه بین هر ستون در نقشه بازی 25 پیکسل و بین هر سطر 24.9 پیسکل فاصله در نظر گرفته شده است که برای محاسبه نگاشت نقشه به ماتریس و برعکس باید به آن دقت نمود.

# توابع مورد استفاده

در این بخش بر روی توابعی که برای هدفی خاص طراحی شده اند بحث خواهیم کرد.

## تابع Wincheck

این تابع در برنامه اصلی نوشته شده است و وظیفه بررسی شرایط برد را بر عهده دارد. بدیهی است که برد زمانی رخ می دهد که تمام خوراکی های معمولی خورده شده باشد. پس از خوردن یک خوراکی عادی همواره این تابع فراخوانی می شود تا شرایط برد بررسی شود. پارامتر این تابع نقشه بازی می باشد که یک ماتریس 19 \* 22 می باشد و یک مقدار false/true بر می گرداند.

bool winCheck(int map[22][19])

{

    for (size\_t i = 0; i < 22; i++)

    {

        for (size\_t j = 0; j < 19; j++)

        {

            if (map[i][j] == 0)

            {

                return false;

            }

        }

    }

    return true;

}

## تابع setmap

این تابع برای چینش نقشه مورد استفاده قرار می گیرد. محل قرارگیری دیوارها و خوراکی های عادی و قدرتی در این تابع مشخص می شود. برای پر کردن نقشه بازی از جدول 1 استفاده شده است.

در این تابع ابتدا، تمام نقاط به 0(خوراکی عادی) مقدار دهی می شود، سپس دیوارها و نقاط قدرتی و موقعیت های خالی مشخص می شود.

    for (size\_t i = 0; i < 22; i++)

    {

        for (size\_t j = 0; j < 19; j++)

        {

            map[i][j] = 0;

        }

    }

در قطعه کد زیر دیوار سمت بالای نقشه مشخص شده است.

for (size\_t i = 0; i < 19; i++)

    {

        map[0][i] = 2;

    }

در قطعه کد زیر نیر یک خانه از نقشه به عنوان دیوار مشخص شده است.

map[1][9] = 2;

## تابع drawmap

این تابع دو پارامتر دریافت می کند، پارامتر اول نام پنجره ای که ترسیمات بر روی آن انجام می شود را دریافت می کند و پارامتر دوم یک ماتریس 19\*22 می باشد که نقشه بازی را تداعی می کند. بخشی از کد که خوراکی های عادی را ترسیم می کند را در قسمت زیر مشاهده می کنیم. برای ترسیم خوراکی های عادی از یک دایره زرد رنگ به شعاع 2 پیکسل استفاده شده است.

 if (map[i][j] == 0)

{

   CircleShape c(2);

   c.setFillColor(Color::Yellow);

   c.setPosition(Vector2f(x, y));

   b.draw(c);

}

برای ترسیم خوراکی های قدرتی نیز از یک دایره سفید توپر به شعاع 5 پیکسل استفاده شده است.

else if (map[i][j] == 1)

{

   CircleShape c(5);

   c.setFillColor(Color::White);

   c.setPosition(Vector2f(x, y));

   b.draw(c);

}

دیوارها در زمان ترسیم مورد استفاده قرار نمی گیرند اما برای جلوگیری از حرکت روح ها و پک-من استفاده می شوند. اگر دیوار ها با مستطیلی به عرض و ارتفاع 10 ترسیم شوند محیط بازی به شکل زیر خواهد بود.



شکل ‏5‑1 ترسیم بازی همراه با دیوارها

این دیوارها در زمان بازی قابل رویت نیستند و برای زیبا تر شدن محیط بازی از ترسیم آن ها جلوگیری شده است.

در این تابع بین هر دو ستون 25 پیکسل و بین هر دو سطر 24.9 پیکسل فاصله می باشد. این اعداد از این جهت در نظر گرفته شده اند که با این اعداد انطباق بیشتری در نگاشت ماتریس به نقشه بازی اتفاق می افتد.

# کلاس های مورد استفاده

## کلاس ghost

این کلاس برای مشخص کردن روح ها در بازی مورد استفاده قرار می گیرد. ویژگی های مورد استفاده برای توصیف یک روح به شرح زیر می باشد:

* مختصات افقی و عمودی : مختصات افقی و عمودی برای تعیین موقعیت روح در نقشه بازی مورد استفاده قرار می گیرد.
* اسپرایت برای حالت عادی ، ترسیده و چشم : برای وضعیت های مختلف روح (عادی – ترسیده – خورده شده) نیز اسپرایت های مختلف مورد استفاده قرار گرفته است.
* سرعت : حالت های مختلف روح سرعت های مختلفی دارد. برای توصیف سرعت حرکت روح از این متغیر استفاده می شود.
* جهت حرکت : جهت حرکت روح را مشخص می نماید. در جدول ‏6‑1 مقادیر درنظر گرفته شده مشخص شده است.

|  |  |
| --- | --- |
| عدد | جهت حرکت |
| 1 | حرکت به سمت راست |
| 2 | حرکت به سمت پایین |
| 3 | حرکت به سمت چپ |
| 4 | حرکت به سمت بالا |

جدول ‏6‑1 جهت حرکت

* وضعیت : یک روح در چند وضعیت می تواند قرار بگیرد که توسط یکی از اعداد جدول ‏6‑2 مشخص می شود.

|  |  |
| --- | --- |
| عدد | وضعیت |
| 1 | روح در حالت پرسه زنی |
| 2 | روح در حالت تعقیب |
| 3 | روح در حالت ترسیده و رنگ آبی |
| 4 | روح در حالت ترسیده و رنگ سفید |
| 5 | روح خورده شده |

جدول ‏6‑2 وضعیت روح

## تابع getStatus

برای دریافت وضعیت روح مورد استفاده قرار می گیرد.

## تابع setStatus

برای تعیین وضعیت روح مورد استفاده قرار می گیرد.

## تابع draw

برای ترسیم اسپرایت روح بر روی نقشه بازی مورد استفاده قرار می گیرد. با توجه به وضعیت روح یکی از تصاویر را برای روح در نظر گرفته و ترسیم می کند.

## تابع setPosition

این تابع برای حرکت روح به مختصات مشخص مورد استفاده قرار می گیرد. دو پارامتر برای مختصات افقی و عمودی دریافت نموده و به مختصات روح اختصاص می دهد.

1. Pac-man [↑](#footnote-ref-1)
2. Simple and Fast Multimedia Library [↑](#footnote-ref-2)
3. کلید W برای حرکت به بالا ، کلید A برای حرکت به چپ ، کلید S برای حرکت به پایین و کلید D برای حرکت به سمت راست مورد استفاده قرار می گیرد. [↑](#footnote-ref-3)
4. در واقع در بازی خوردن اتفاق نمی افتد و با رسیدن پک- من به مختصات هر نقطه فرض می کنیم پک-من آن نقطه را خورده و دیگر آن نقطه نمایش داده نمی شود. و امتیاز آن نقطه را برای بازیکن منظور می کنیم. [↑](#footnote-ref-4)
5. زمان در نظر گرفته شده ، مطابق جدول ارائه شده در زمان بندی پروژه برای مرحله اول می باشد. [↑](#footnote-ref-5)
6. تعداد در نظر گرفته شده مطابق جدول ارائه شده در جدول زمان بندی پروژه می باشد. [↑](#footnote-ref-6)
7. توضیحات تکمیلی در بخش مجزایی در قسمت کدها خواهد آمد. [↑](#footnote-ref-7)
8. اسپرایت های مورد استفاده از سایت <https://tcrf.net/Pac-Man_%28Arcade%29> استخراج شده اند. [↑](#footnote-ref-8)
9. عکس مورد استفاده برای نقشه عرض 451 پیکسل و ارتفاع 500 پیکسل دارد. برای نگاشت بهتر، مقیاس این عکس در پروژه کمی افزایش یافته است. [↑](#footnote-ref-9)