# پـروژهٔ پـایانی درس مبـانی کامپـیوتر؛

# بـازی Snake and Food

**کد و ارائه: محمدپویا تراشی**

**استاد محترم: دکتر داودآبادی**

**بهمن ۱۴۰۱**

**مقدمه**

بازی محبوب Food vs Snake که اغلب با نام "مارِ نوکیا"یا "Matopeli" شناخته می‌شود، از شناخته‌شده ترین بازی‌های ژانر آرکید است. هدف این پروژه پیاده‌سازی این بازی با استفاده از کتابخانهٔ PyGame می‌باشد.

## قوانین و منطق بازی

در ابتدای این بازی، ماری به طول واحد در صفحه قرار دارد. با شروع بازی، مار به حرکت در می‌آید و حرکت آن به‌وسیلهٔ کلیدهای جهت‌نمای کیبورد و یا WASD قابل کنترل است. با برخورد مار با دیواره‌ها یا موانع موجود در مسیر، بازی به پایان می‌رسد و این به معنای باخت بازیکن است. در طول بازی رنگ برخی خانه‌ها تغییر می‌کند که نشان‌دهندهٔ وجود غذا در آن خانه‌هاست. با عبور مار از این خانه‌ها، یک واحد به طول مار اضافه می‌گردد. مسیر حرکت هر خانه از مسیر حرکت اولین خانه (سرِ مار) پیروی می‌کند. با خورده شدن هر غذا، خانه ای دیگر به صورت رندوم انتخاب شده و رنگ آن تغییر می‌کند، به عبارتی دیگر، غذا در خانه‌ای دیگر قرار می‌گیرد.

## روش حل

در چندین مرحله به بررسی مسیر اجرای پروژه می‌پردازیم؛

**۱. طراحی صفحهٔ بازی:** صفحهٔ بازی به تعداد زیادی سلول تقسیم می‌گردد. طول هر سلول تعداد پیکسل معینی‌ست. به بیانی دیگر، صفحه شبیه یک جدول با تعدادی معین خانهٔ افقی و تعدادی معین خانهٔ عمودی‌ست. قرارگیری اجزا در این صفحه بر اساس همین جدول اتفاق می‌افتد. بعنوان مثال مار در ابتدا در خانهٔ ۵ افقی و ۱۰ عمودی نمایش داده می‌شود. با خوردن یک سیب و افزوده شدن یک واحد به طول مار، خانه‌های (۵ و ۱۰) و (۴و ۱۰) نشان‌دهندهٔ مکان مار خواهند بود. بر همین اساس، مکان غذاها نیز تعیین می‌گردد.

۲. **بدن و جهت حرکت مار:** بدن مار به صورت لیستی در زبان پایتون است. این لیست شامل مختصات تک تک خانه‌های بدن مار است. با خوردن هر واحد غذا، یک عضو به اعضای این مجموعه اضافه می‌گردد که به معنای افزایش طول مار است. همچنین یک بردار دیگر با نام جهت حرکت برای مار تعریف می‌شود که در هر حرکت مار به مختصات سر مار اضافه می‌گردد. مقداری پیشفرض دارد و مقدار آن با زدن کلیدهای جهت‌نما تغییر می‌یابد.

| action | direction |
|---|---|
| Snake moves right in every step | **(1, 0)** |
| Snake moves left in every step | **(-1, 0)** |
| Snake moves up in every step | **(0, -1)** |
| Snake moves down in every step | **(0, 1)** |

۳. **حرکت مار:** در هر واحد حرکت مار، لیست شامل مختصات بدن مار به این شکل تغییر می‌کند:

- آخرین عضو لیست حذف می‌شود،
- کپی‌ای از مقدار اولین عضو لیست با بردار جهت حرکت جمع می‌شود و به لیست افزوده می‌شود.

۴. **قرارگیری و خورده شدن غذا:** برای هر واحد غذا مختصاتی به صورت رندوم درنظر گرفته می‌شود و سلول مربوطه رنگی می‌شود. با برابر شدن مختصات سر مار و غذا، یک واحد به لیست بدن مار افزوده می‌گردد و تابع رسم غذا در مکان رندوم، مجددا فراخوانی می‌شود.

۵. **حالات باخت:**

- در صورتی که موقعیت سرِ مار با موقعیت سلول‌های مرزی صفحه برابر شود، باخت بازی‌کن رقم می خورد.
- در حالتی که مختصات سر مار برابر با مختصات یکی دیگر از بلوک‌های بدنش شود، حالت باخت رخ می‌دهد.
- در حالتی که موقعیت سر مار با موقعیت دیواره ها برابر شود، باخت رخ می‌دهد.

**متن فوق منطق حل مسئلهٔ اصلی بازی‌ست که بوسیلهٔ پایتون و کتابخانهٔ پای‌گیم پیاده‌سازی و اجرا شده است. توابع، کلاس‌ها، متغیرهای مورد استفاده در ارائه بررسی خواهند شد.**

**مخزن زیر، شامل مراحل انجام پروژه است.**

**https://github.com/PooyaTarashi/Nokia-Snake**

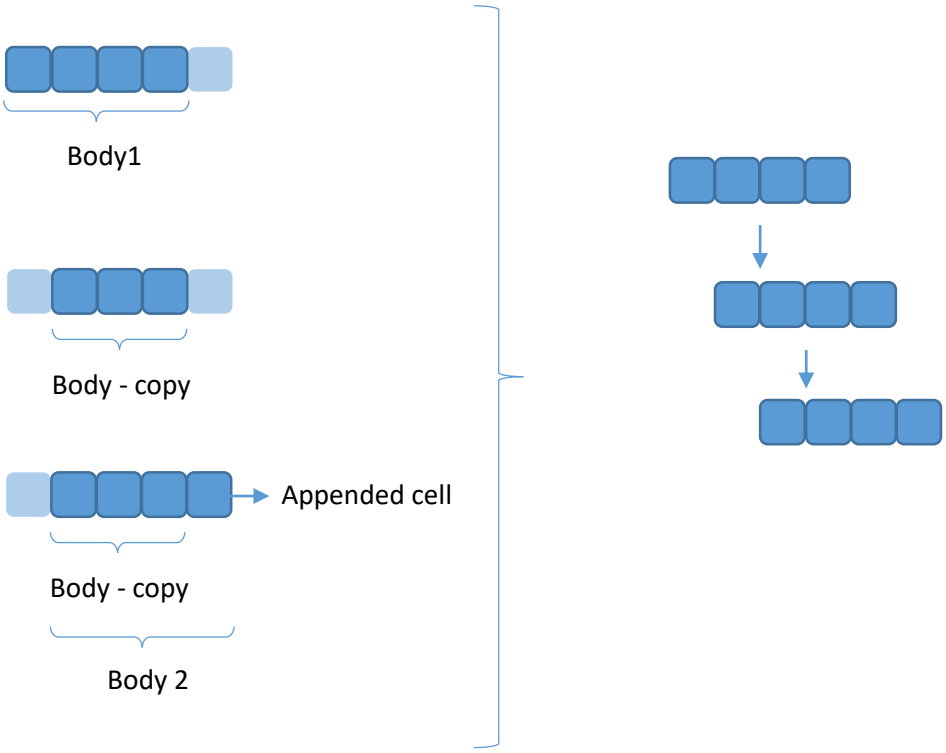
## بررسی پیاده سازی کد

برنامه شامل سه تابع اصلی با نام‌های main_menu، game_screen و game_over_menu، و سه کلاس با نام‌های ای SNAKE، FOOD و BARRIER است. هر تابع شامل متغیر screen است که به کمک یک حلقهٔ while و دستور display.update() صفحه را باز نگه می‌دارد و محتویات آن را آپدیت می‌کند.

**محتویات و عملکرد تابع main_menu**: شامل یک لوپ است که رویدادهای ممکن را بررسی می‌کند. دو رویداد اصلی‌ای که برای آن تعریف شده، زدن کلید بسته شدن پنجره است که با استفاده از دستور sys.exit() کلیهٔ کدهای اجرایی را می‌بندد و صفحه بسته می‌شود.

رویداد دیگر، زدن یکی از کلیدهای اینتر یا اسپیس است که در اثر آن یک تایمر روی صفحه چاپ می‌شود و شمارش معکوس تا شروع بازی را با استفاده از دستور pygame.time.delay(800) انجام می‌دهد. پارامتر ورودی ۸۰۰ به معنای تاخیر ۸۰۰ میلی‌ثانیه‌ای برای شمارش است. پس از اتمام شمارش معکوس، تابع game_screen فراخوانی می‌گردد. قبل از بررسی این تابع، به تشریح کلاس‌ها پرداخته می‌شود.

**کلاس SNAKE**: شامل سه تابع اصلی‌ست. اولین تابع (__init__) لیست شامل بلاک‌های بدن مار و جهت حرکت دیفالت آن را تعریف می‌کند. تابع دوم (darw_snake) شامل یک حلقهٔ for است که سلول‌های مربوط به لیستِ بدنِ مار را به‌وسیلهٔ دستور draw rectangle نمایش می‌دهد. سومین تابع (move_snake) حرکت مار را انجام می‌دهد. نحوهٔ عملکرد آن همان‌طور که در بخش روش‌ها گفته شد، بدین شرح است. ابتدا یک کپی موقت از لیست بدن مار می‌گیرد اما آخرین عضو لیست در این کپی وجود ندارد. سپس با دستور append به آخرین عضو لیستِ بدن، به اندازهٔ بردار جهت حرکت اضافه می‌کند و آن را به لیست اضافه می‌کند. (بدون این‌که بلاک سرِ مار را از لیست حذف کند.) در ادامه مقدار لیستِ body_copy را در لیست body اصلی کپی می‌کند.



**کلاس FOOD**: عملا شامل دو تابع اصلی‌ست. اولین تابع (draw_food) با توجه به موقعیت و رنگ غذا، آن rectangle مربوط به آن را screen رسم می‌کند. دومین تابع (reinitialize_random_position) یک پوزیشن رندوم برای محل قرارگیری غذا درنظر می‌گیرد. در عین حال بررسی می‌کند که این موقعیت توسط موانع پوشیده نشده باشد.

**کلاس BARRIER**: توابع مربوط به آن شبیه توابع مربوط به کلاس FOOD است. با این تفاوت که چون دیوارها در طول بازی فقط یک بار ساخته می‌شوند، نیازی به تابع برای ساخت مجدد آن‌ها وجود ندارد. پس پوزیشن آن‌ها در همان تابع initiation ساخته می‌شود. تابع دوم که وظیفهٔ رسم را بر عهده دارد، سلول‌های مربوطه را رنگ‌آمیزی می‌کند.

**تابع game_screen**: تمامی اتفاقات بازی در همین تابع رخ می‌دهد. در اولین مرحله یک آبجکت برای هر کلاس در نظر گرفته می‌شود. سپس لیستی شامل موقعیت موانع ساخته می‌شود که از تابع رندوم استفاده می‌کند. در ادامه، در حلقهٔ وایل بازی، شبیه همان چیزی که در تابع main اتفاق افتاد، بررسی برای دریافت کلیدها انجام می‌شود. با زدن کلیدهای جهت‌نما، بردار جهت حرکت مار تغییر می‌کند. سپس توابع مربوط به آبجکت‌ها در هر بار رفرش شدن صفحه فراخوانی می‌شوند تا حرکت مار، رسم غذاها، رسم دیوارها و ... انجام گردد.

بررسی می‌گردد اگر مختصات سر مار با مختصات غذا یکی شد، یک واحد از جهت درست به بدن مار اضافه می‌گردد و موقعیت غذا دوباره به شکل رندوم تعریف می‌شود.

در ادامه به کمک یک فایل txt، حداکثر رکورد بررسی می‌شود. همچنین یک بلاک برای نوشتن امتیاز بازیکن ساخته می‌شود.

آخرین بخش، بررسی حالات باخت بازیکن است. بازیکن در سه حالت به صفحهٔ آخر منتقل می‌شود.

- مختصات سرِ مار در لیست مربوط به مختصات موانع وجود داشته باشد.
- مختصات سرِ مار از رنج صفحه خارج شود.
- مختصات سر مار دوبار در لیست تکرار شود که به معنی برخورد آن با بدن خودش است.

**تابع game_over_screen**: آخرین بخش بازی‌ست و ساز و کاری مشابه سایر توابع اصلی دارد. با زدن کلید اینتر به صفحهٔ اول باز می‌گردد و با زدن x کلیهٔ کدها متوقف می‌شوند.