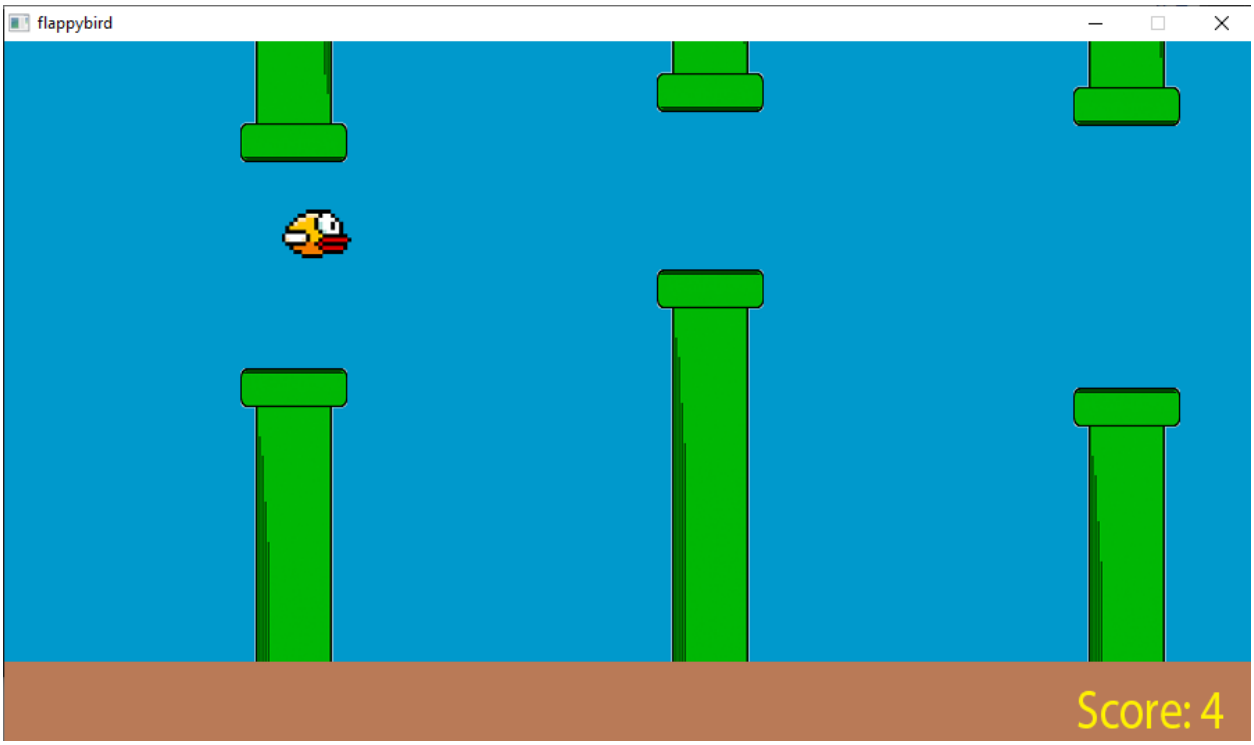

بازی Flappy Bird

پروژه‌ی درس برنامه‌سازی پیشرفته

مهدی امیری شوکی ۹۸۵۲۲۱۴۸

استاد درس: دکتر موحدی

فهرست

۱.مقدمه

در این پروژه، بازی مشهور Flappy bird در زبان C++ بصورت شی‌گرا و به کمک کتابخانه‌ی گرافیکی SDL پیاده‌سازی شده است. بازی بطور کامل روی Windows 10 نوشته و تست شده است ولی احتمالا به شرط اجرای نکات گفته‌شده در فایل README.md با دیگر نسخه‌های Windows نیز سازگار است و مشکلی در اجرا نداشته باشد. علاوه بر شی‌گرایی از مفاهیم بازتعریف عملگرها (Operator overloading)، اشاره‌گرها و مقادیر رفرنس (Reference variables and pointers) نیز استفاده شده است. در گزارش حاضر به توضیح بازی، تشریح روند ساخت بازی در چندین بخش و همچنین به موارد آموزشی مفید این پروژه خواهیم پرداخت.

۲.توضیح اجمالی روند بازی

در بازی مشهور Flappy Bird یک پرنده به عنوان شخصیت اصلی بازی تحت تاثیر جاذبه زمین قرار دارد اما خود نیز توانایی جهیدن با شتاب $\frac{۵}{۴}$ برابر شتاب گرانش زمین در جهت مخالف را دارد. هدف بازیکن رد کردن پرنده از بین لوله‌هایی است که در موقعیتی تصادفی از بالا و پایین صفحه بیرون زده‌اند. بازیکن می‌تواند با فشردن کلید Spacebar در یک لحظه از زمان، شتاب رو به بالا با کیفیت گفته‌شده ایجاد کند تا اولا از سقوط پرنده جلوگیری و ثانیا تلاش کند پرنده را از میان لوله‌ها که به عنوان مانع سر راه او هستند رد کند.[۱]

[۱] www.flappybird.io

۳. توضیح کد

۱.۳. کلاس Bird

مختصات پرنده، سرعت در راستای محور y و امتیاز پرنده در شی پرنده ذخیره می‌شوند. شی پرنده قادر است فایل bmp تصویر پرنده را برای استفاده بارگذاری و در خود ذخیره کند، پرش انجام دهد و خلاف جهت گرانش زمین شتاب بگیرد، با استفاده از بازتعریف عملگر += جایگاه خود در صفحه را تصحیح کند، روی صفحه‌ی بازی و در جایگاه مناسب تصویر پرنده را نمایش دهد و امتیاز نمایش‌داده‌شده روی صفحه را به مقدار مناسب بروز کند.

Bird
- x: int - y: int - s: int - score: int - image: SDL_Surface*
+ Bird(Graphics&) + jump() + operator+=(const int) + blit(Graphics&) + updateScore(Hurdle*, Score&, Graphics&)

نمودار UML ۱. کلاس Bird

*در محاسبه‌ی جایگاه پرنده از معادله‌ی مکان-زمان استفاده شده است.

معادله‌ی مکان - زمان: $x(t) = tv + x_0$

۲.۳. کلاس Graphics

صفحه‌ی بازی، سطح (Surface) نمایش تصاویر روی صفحه‌ی بازی و رویدادهای حین بازی (Event) در شی Graphics ذخیره می‌شوند. این شی می‌تواند صفحه‌ی بازی را به طول و عرض موردنیاز بسازد، تصویر ورودی را در مختصات ورودی با طول و عرض ورودی نمایش دهد، صفحه‌ی بازی را پس از تغییرات انجام شده بروز کند، در برنامه به مقدار زمان موردنیاز در ورودی توقف زمانی ایجاد کند، صفحه‌ی بازی را ببندد، رویدادهای ورودی را دریافت کند، بررسی کند برنامه به پایان رسیده یا نه و یک فایل تصویری با فرمت bmp را بارگذاری کند.

Graphics
- window: SDL_Window* - screenSurface: SDL_Surface* - event: SDL_Event
+ Graphics(const int, const int) + blit(SDL_Surface*, const int, const int, const int, const int) + update() + delay(int) + endGame() + getEvents() + isClosed() + loadImage(string)

نمودار UML ۲. کلاس Graphics

۳.۳. کلاس Hurdle

موقعیت مکانی لوله‌های یک مانع و فاصله‌ی روزنه‌ی باز بین دو لوله در شی مانع ذخیره می‌شوند. شی مانع می‌تواند مقادیر تصادفی موقعیت مکانی لوله‌ها و فاصله‌ی بین دو لوله را با استفاده از کتابخانه‌های ctime و cstdlib تولید کند، دو لوله‌ی مانع را روی صفحه‌ی بازی نمایش دهد، با گذشت زمان مقدار مختصه‌ی افقی مانع را افزایش دهد، برخورد (Collision) بین هریک از لوله‌های مانع و پرنده را مشخص کند و مختصه‌ی افقی مانع را گزارش کند.

Hudle
- x: int - y: int - h: int
+ Hurdle() + blit(Graphics&, SDL_Surface*, SDL_Surface*) + setX(int) + intersect(int, int, int, int) + getX2()

نمودار UML ۳. کلاس Hurdle

۴.۳. کلاس Score

یک شی به نام Score در هر لحظه از بازی مقدار عددی امتیاز بازیکن را در خود ذخیره می‌کند، به‌علاوه این شی تصاویر مربوط به نمایش امتیاز بازی را هم در خود ذخیره شده دارد تا بسته به امتیاز بازیکن هر یک از آن‌ها را که نیاز است روی صفحه، نمایش دهد.

Score
- score: int - images: SDL_Surface* [NUMOFHURDLES + 1]
+ Score(Graphics&) + setScore(int) + blit(Graphics&)

نمودار UML ۴. کلاس Score

۵.۳. تابع main

از کلاس‌های Bird، Score، Graphics یک شی و از کلاس Hurdle ۲۰ شی (به اندازه‌ی ثابت NUMOFHURDLES) در main ساخته و ذخیره می‌شوند.
حلقه‌ی اصلی برنامه (mainloop) در main قرار دارد و در هر بار اجرای حلقه بررسی می‌شود که به یکی از سناریوهای پایان برنامه یعنی برد یا باخت رسیده‌ایم یا خیر. به‌علاوه، ورودی کاربر هم با استفاده از متد getEvents از شی Graphics بررسی می‌شوند و در صورت‌نیاز عملگر =+ برای شی کلاس Bird فراخوانی می‌شود تا موقعیت پرنده را تغییر دهد.

وظیفه‌ی دیگر حلقه‌ی اصلی حرکت دادن موانع نسبت به پرنده است تا این‌طور به نظر بیاید که پرنده در راستای محور طول‌ها حرکت دارد و این حرکت با تغییر مختصه‌ی افقی تک‌تک شی‌های مانع به کمک تابع setHurdlesX صورت می‌گیرد.

جهت جلوگیری از گیرکردن برنامه و به‌اصطلاح not responding شدن و از فرمان خارج‌شدن صفحه‌ی برنامه، در هر بار اجرای حلقه‌ی اصلی با استفاده از متد delay از شی Graphics تاخیر زمانی ۵۰ میلی‌ثانیه‌ای ایجاد می‌شود، همچنین امکان خروج از برنامه با کلیک روی دکمه‌ی escape پنجره‌ی برنامه تا قبل از رسیدن به یکی از سناریوهای پایانی بازی وجود دارد ولی اگر بازیکن بازی را ببرد یا ببازد به مدت ۱۰ ثانیه پیغام مربوط به برد یا باخت نمایش و سپس خود به خود برنامه بسته خواهد شد و نیاز به اقدام کاربر نخواهد بود که این مسئله هم با متد getEvents کنترل می‌شود و همانطور که گفته شد با رسیدن به سناریوی پایان، برنامه از حلقه‌ی اصلی خارج و با استفاده از متد delay ۱۰ ثانیه صبر و سپس با متد endgame از شی Graphics، پنجره‌ی بازی را می‌بندد.

۶.۳. ثوابت (Constants)

مقدار	نام مورد استفاده در کد	نام یا توضیح ثابت
۹۰۰	SCREENWIDTH	عرض صفحه‌ی بازی
۵۰۴	SCREENHEIGHT	طول صفحه‌ی بازی
۵۰	BIRDWIDTH	عرض تصویر پرنده
۳۵	BIRDHEIGHT	طول تصویر پرنده
۷۷	PIPEWIDTH	عرض تصویر لوله
۳۰۰	PIPEHEIGHT	طول تصویر لوله
۱۳۲	SCOREWIDTH	عرض تصویر امتیاز
۴۸	SCOREHEIGHT	طول تصویر امتیاز
۲۰۰	INITX	طول موقعیت اولیه پرنده
۱۵۰	INITY	عرض موقعیت اولیه پرنده
۴۴۶	GROUND	عرض زمین
۲۰	NUMOFHURDLES	تعداد موانع
۹۰۰	FIRSTHURDLEX	طول اولین مانع در مبدا زمان
۳۰۰	HURDLESDISTANCE	فاصله‌ی دو مانع در محور x
۱۰	TIMESPEED	سرعت پرنده نسبت به موانع در محور x
۱۰	VG	سرعت حدی پرنده در محور y
۱۲–	VU	حداکثر سرعت ایجاد شده‌ی پرش در محور y
۱۰۰	MINH	کمترین مقدار فاصله‌ی بین دو لوله در مانع
۲۵۰–	MINY	کمترین مقدار عرض لوله‌ی بالایی در مانع
۱۰۰	DOMAINH	دامنه‌ی تولید اعداد تصادفی

جدول ۱. ثوابت

۴. لینک به فایل اجرایی برنامه

برای اجرای بازی در اینجا کلیک کنید

- در صورتی که با خطای SDL2.dll is missing مواجه شدید دستورالعمل موجود در فایل README.md را دنبال کنید.

۵. آموخته‌های مفید پروژه

- کار با git برای ذخیره و نگه‌داری نسخه‌های پروژه بصورت خصوصی (private)
- کار با کتابخانه‌ی گرافیکی SDL
- یادگیری راه و چاه آماده‌سازی فایل نهایی برنامه‌ی نوشته‌شده با کتابخانه‌ی SDL برای اجراشدن روی کامپیوترهایی به غیر از کامپیوتر برنامه‌نویس
- تجربه‌ی نوشتن برنامه‌ای که قوانین ساده‌ی فیزیک سقوط آزاد را در محیطی دوبعدی شبیه‌سازی کند.
- تجربه‌ی جست‌وجو و ساخت منابع تصاویری موردنیاز برای استفاده در یک پروژه‌ی بازی‌سازی
- تجربه‌ی نوشتن گزارش کار برای یک پروژه‌ی برنامه‌سازی