برنامه‌سازی پیشرفته

تمرین کامپیوتری شماره ۵

استاد: **دکتر رامتین خسروی**

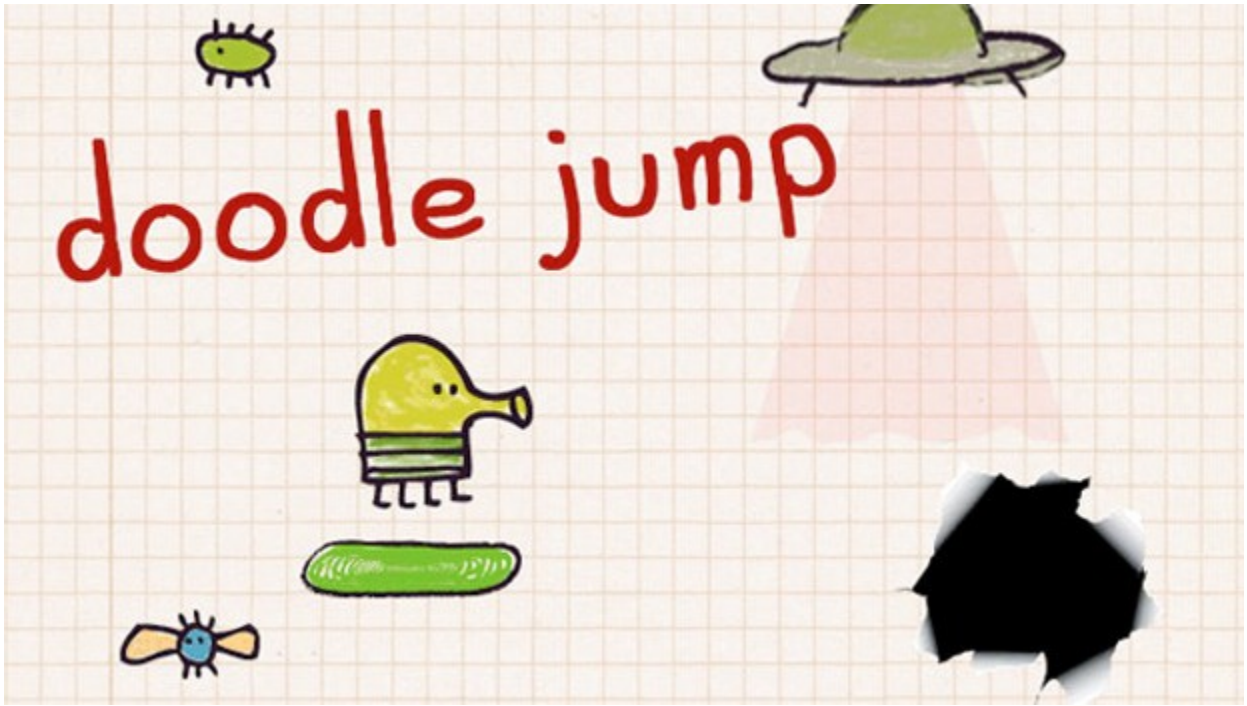
طراحان: علی زارع، مهیار کریمی، کیمیا محمدطاهری، سهیل ذیبخش

مهلت تحویل: **جمعه ۲۴ اردیبهشت ۱۴۰۰، ساعت ۲۳:۵۵**

مقدمه

هدف از این تمرین آشنایی شما با برنامه‌نویسی شیءگرای رویدادمحور[1] و استفاده از آن در کنار کتابخانه‌های گرافیکی است. انتظار می‌رود از تکنیک‌های برنامه‌نویسی که تاکنون در کلاس درس فراگرفته‌اید یا در هنگام تحویل حضوری تمرین‌ها به شما تذکر داده شده است به طور کامل در این تمرین استفاده کنید.

برای پاسخ به این تمرین باید از کتابخانه‌ی گرافیکی SDL[2] استفاده کنید. برای راحتی کار شما، در این لینک یک کتابخانه‌ی واسط به نام RSDL[3] برای کار کردن با SDL در اختیار شما قرار داده شده است. قبل از شروع به انجام این تمرین توصیه می‌شود حتماً ویدیوهای مربوط به برنامه‌نویسی رویداد محور را ببینید و مستندات موجود در این لینک را مطالعه کنید.



[1] Event Driven Programming

[2] Simple DirectMedia Layer

[3] Ramtin Simple DirectMedia Layer

Doodle Jump!

بازی Doodle Jump از بازی‌های سبک پلتفرمر است. برای آشنایی بیشتر با این بازی و تجربه آن می‌توانید به این لینک مراجعه کنید.

دقت کنید بازی موجود در لینک به طور کامل با این تمرین یکسان نیست و حتماً جزئیات پروژه را در صورت تمرین مطالعه کنید.

روند بازی

در این بازی، بازیکن یک کاراکتر را کنترل می‌کند که روی سکوهای موجود در صفحه می‌پرد و بالا می‌رود. هدف این است که تا جای ممکن بالا برویم و در نتیجه حداکثر امتیاز ممکن را کسب کنیم. امتیاز بازیکن برابر با حداکثر ارتفاعی (بر حسب پیکسل) است که نقطه‌ی پایینی بدن کاراکتر به آن رسیده است. هر قدر که کاراکتر بازی بالاتر می‌رود سکوهای جدید نمایان می‌شوند و تنها هنگامی بازی تمام می‌شود که بازیکن ببازد؛ یعنی یا کاراکتر به دشمنی برخورد کند یا سر کاراکتر از پایین صفحه پایین‌تر برود.

گرچه کاراکتر همیشه به سمت بالا می‌رود، اما هیچ‌وقت از بالای صفحه خارج نمی‌شود. اگر پای کاراکتر از وسط صفحه خواست بالاتر برود سکوهای جدیدی نمایان می‌شود. هنگامی که کاراکتر می‌خواهد از وسط صفحه بالاتر برود دوربین هم با کاراکتر حرکت میکند. همچنین امتیاز فعلی بازیکن باید در صفحه نشان داده شود.

توجه کنید که جدولی حاوی اندازه‌های مربوط به بازی به شما داده شده است. داده‌هایی که در جدول وجود ندارند باید به تصمیم خودتان به صورتی انتخاب شوند که بازی قابل اجرا باشد و مقدار دقیق مهم نیست.

حرکت‌های کاراکتر

- کاربر فقط با کلیدهای A و D با کاراکتر ارتباط برقرار می‌کند. با کلید A به سمت چپ و با کلید D به سمت راست میرویم.
- توجه کنید که اگر کاراکتر به سمت راست حرکت می‌کند باید جهت سرش سمت راست باشد و بالعکس. هنگامی که حرکت افقی نمی‌کند جهت سرش آخرین جهتی که در حال حرکت به سمت آن بود باشد.
- حرکت افقی کاراکتر شتاب‌دار نیست یعنی با زدن کلید و برداشتن کلید کاراکتر سرعت افقی ثابتی می‌گیرد.

- کاراکتر در حالت سقوط آزاد است تا این که به یک سکو برخورد کند و دوباره به بالا بپرد. هرگاه پای کاراکتر هنگام پایین آمدن به بالای سکو برخورد کند به سمت بالا می‌پرد. بنابراین کاراکتر هنگام بالا رفتن با سکو برخورد ندارد و تنها وقتی برخوردی صورت می‌گیرد که کاراکتر روی سکو فرود بیاید.
- اگر کاراکتر از مرز چپ یا راست صفحه بیرون برود از سمت دیگر وارد می‌شود.
- مکان اولیه‌ی کاراکتر از وسط صفحه شروع می‌شود و در حال به بالا پریدن است.
- پیاده‌سازی انیمیشن جمع شدن پاهای کاراکتر هنگام پرش نمره‌ی **امتیازی** دارد.

سکوها

در این پیاده‌سازی از بازی سه نوع سکو وجود خواهد داشت:

- **سکوی عادی**: این سکوها کاملا ثابت و صلب هستند. کاراکتر هنگامی که روی این سکوها فرود می‌آید، با سرعت اولیه ثابتی به سمت بالا می‌رود.
- **سکوی متحرک**: این سکوها دائما درحال حرکت افقی هستند و از سمت چپ صفحه به راست و برعکس حرکت می‌کنند. برخورد کاراکتر با این سکوها مانند سکوهای عادی است، یعنی با فرود بر روی آن‌ها کاراکتر می‌پرد. می‌توانید فرض کنید این سکوها در شروع حرکت‌شان همیشه به سمت راست حرکت می‌کنند و با رسیدن به مرز راست صفحه، تغییر جهت می‌دهند و تا مرز چپ صفحه حرکت می‌کنند تا تغییر جهت دهند و همواره این رفتار را ادامه می‌دهند.
- **سکوی شکننده**: این سکوها ظاهری شکننده دارند و هنگامی که کاراکتر روی آن‌ها فرود می‌آید درجا می‌شکنند و کاراکتر سرعت رو به بالا نخواهد گرفت. در واقع این سکوها کمکی به بازیکن نمی‌کنند. پیاده سازی انیمیشن شکستن این سکوها نمره‌ی **امتیازی** دارد.

دشمن

ما فقط یک مدل دشمن داریم و آن هم نوع دشمن ثابت است. این دشمن‌ها در یک‌جا از دنیای بازی ثابت ایستاده‌اند و هنگامی که هرگونه برخوردی با آن‌ها داشته باشیم بازی تمام می‌شود. برای اندازه و تصویر دشمنان هر مقدار و عکسی از فایل‌های داده‌شده که منطقی باشد می‌توانید استفاده کنید.

فنر

فنرها در دنیای بازی ثابت هستند و هنگامی که روی یک فنر بپریم، سرعت پرش بیشتری پیدا می‌کنیم (حدود ۲ برابر) و در نتیجه در همان لحظه پرش بلندتری خواهیم داشت.

نقشه بازی

برای تولید نقشه بازی دو روش وجود دارد: تولید نقشه ثابت و تولید نقشه داینامیک. در ادامه هر کدام توضیح داده شده اند و شما باید فقط یکی از این دو روش را انتخاب و پیاده سازی کنید. تولید نقشه داینامیک امتیازی است.

- **نقشه ثابت:**

در این قسمت شما باید اطلاعات مختصات هر عنصر بازی را از فایل map.txt که در کنار برنامه شما قرار می‌گیرد بخوانید و در مختصات مربوط به آن قرار دهید، فرمت فایل ورودی به شکل زیر است:

```
<number of entities>
    <x> <y> <type>
    <x> <y> <type>
    ...
```

در خط اول تعداد کل موجودیت های نقشه آمده است و سپس در ادامه در هر خط مختصات نقطه‌ی وسط پایین هر موجودیت و نوع آن داده می‌شود. این مختصات نشان‌دهنده‌ی فاصله‌ی آن موجودیت از نقطه‌ی پایین چپ شروع نقشه است.

برای فهم بهتر یک نمونه در ادامه آمده است:

```
5
320 0 platform
520 0 platform
270 100 mplatform
370 200 platform
520 100 enemy
```

دقت کنید که در این نوع نقشه پس از رسیدن به بالاترین موجودیت بازی، موجودیت جدیدی اضافه نمی‌شود.

- **نقشه داینامیک (امتیازی):**

می‌دانیم که حافظه‌ی کامپیوتر مقدار محدودی دارد و نمی‌توانیم نقشه‌ی بازی را تا ارتفاع بسیار زیادی ذخیره کنیم. همچنین اگر کل نقشه‌ی بازی را از ابتدا مشخص کنیم، هر بار انجام بازی تکراری می‌شود و جذابیت دوباره بازی کردن بسیار کمتر. از این رو نقشه و دنیای بازی همینطور که کاراکتر بالا می‌رود ساخته می‌شود و از قبل پیش‌ساخته نیست. از طرفی، مهم است که هرچه کاراکتر بالاتر می‌رود، بازی کم‌کم سخت‌تر شود.

برای ساخت دنیای چنین بازی‌هایی روش‌های متفاوتی استفاده می‌شود؛ یک روش متداول که کنترل سختی بازی را ساده می‌کند و ما در این پیاده‌سازی از آن استفاده می‌کنیم، استفاده از بلوک‌های پیش‌ساخته است. این روش به این صورت است که ما تعدادی دنباله از موجودیت‌ها برای نقشه بازی در اختیار داریم و با پیشرفت بازی، از بین دنباله‌ها یک دنباله را انتخاب می‌کنیم و ادامه نقشه را به کمک آن می‌سازیم. این دنباله‌ها را با فرمت مشخصی در یک فایل در کنار فایل اجرایی بازی ذخیره شده است.

شما باید نقشه بازی، مکان، نوع سکوها و دشمنان را از روی یک فایل متنی به صورتی که در ادامه گفته می‌شود تولید کنید.

دنباله‌ها

فایل sequence.txt در کنار برنامه شما قرار می‌گیرد و در آن تعدادی دنباله[4] از موجودیت‌های بازی قرار دارد. نمونه‌ای از ساختار این فایل را در زیر مشاهده می‌کنید:

```
<number of sequences>
<start> <end> <total height> <number of entities>
    <x> <y> <type>
    <x> <y> <type>
    …

<start> <end> <total height> <number of entities>
    <x> <y> <type>
    <x> <y> <type>
    …

…
```

[4] sequence

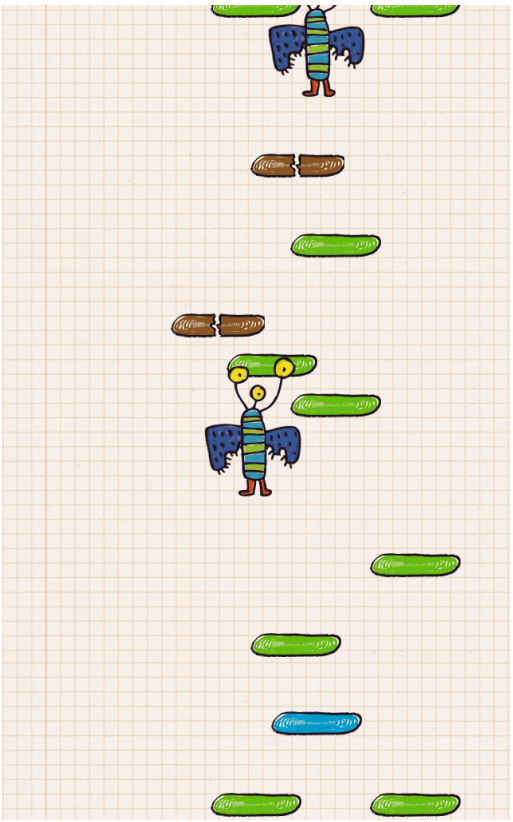
- در خط اول این فایل **تعداد دنباله‌ها** داده شده است و از خط دوم **مشخصات دنباله‌ها** می‌آیند. به ازای **هر دنباله** ابتدا در یک خط *امتیاز شروع آمدن دنباله* و *آخرین امتیازی که دنباله* می‌آید. سپس ارتفاع مطلق سکوهای دنباله و تعداد موجودیت‌های آن دنباله داده می‌شود.
  - توجه کنید که اگر امتیاز پایانی -1 بود به این معنا است که سقفی در نظر نمی‌گیریم.
- در ادامه در هر خط مختصات نقطه‌ی وسط پایین هر موجودیت و نوع آن داده می‌شود. این مختصات نشان‌دهنده‌ی فاصله‌ی آن موجودیت از نقطه‌ی پایین چپ شروع آن دنباله است. مشخصات هر دنباله هم با یک خط خالی از هم جدا شده‌اند.

شیوه‌ی تولید توالی دنباله‌ها به صورت تصادفی است؛ به این صورت که در هنگامی که کاراکتر در بازه‌ی امتیازی چند دنباله بود بین آن دنباله ها به صورت تصادفی یکی را برای ساختن تکه‌ی بعدی استفاده می‌کنیم.

انواع موجودیت‌ها در فایل

	نوع موجودیت	کلید
	سکوی عادی	`platform`
	سکوی متحرک	`mplatform`
	سکوی شکننده	`bplatform`
	فنر	`spring`
	دشمن	`enemy`

ساختار نمونهٔ فایل



```
3
0 5000 300 4
320 0 platform
520 0 platform
270 100 mplatform
370 200 platform

100 5000 400 5
520 0 platform
320 100 enemy
420 200 platform
340 250 platform
270 300 bplatform

300 -1 300 3
420 0 platform
370 100 bplatform
400 200 enemy
```

ابتدا باید توجه کنیم که مبدأ مختصات در این فایل، گوشه‌ی پایین چپ صفحه در نظر گرفته شده است. همچنین به ازای هر موجودیت توصیف شده در فایل ورودی، مختصات وسط و پایین آن نسبت به نقطه‌ی پایین چپ دنباله‌ی فعلی مشخص شده است.

فرض کنیم ابتدا اولین دنباله‌ی توصیف شده از این فایل برای ساخته‌شدن انتخاب می‌شود و سپس دنباله‌ی دوم و بعد دنباله‌ی سوم. در این صورت، شکل بالا پایین‌ترین قسمت نقشه‌ی بازی را نشان می‌دهد.

توجه کنید که سکویی که y آن در فایل برای مثال ۱۰۰ است، در دنیای بازی و در صفحه، y آن می‌تواند متفاوت باشد زیرا این عدد در واقع فاصله‌ی عمودی آن با نقطه‌ی شروع دنباله است که برابر جمع ارتفاع دنباله‌های قبل آن است. در نتیجه به عنوان مثال در عکس بالایی، دشمن بالایی متعلق به دنباله‌ی سوم است که در آن دنباله y آن برابر ۲۰۰ است اما در بازی y آن ۹۰۰ (۳۰۰ + ۴۰۰ + ۲۰۰) است.

پایان بازی

هنگامی که کاراکتر به دشمن برخورد کند یا سر کاراکتر از پایین صفحه پایین تر برود بازی تمام می‌شود. در این زمان باید پیامی مبنی بر تمام شدن بازی همراه با امتیاز بازیکن روی صفحه نمایش داده شود. اگر بازیکن کلیدی را فشار داد بازی بسته می‌شود.

اندازه‌ها

اندازه‌ی اجسام بازی را می‌توانید با توجه با اندازه‌ی پیکسلی فایل‌های عکسی داده شده تنظیم کنید. توجه کنید که تصویر بعضی اجسام (مانند سکوها، فنر و دشمن‌ها) در قالب یک فایل آمده (game-tiles) که باید یا این فایل‌ها را جداگانه کراپ کنید یا با استفاده از مستطیل مبدا در تابع draw_img در rsdl هنگام کشیدن آن‌ها بر روی صفحه قسمت مورد نظر را ببرید. اندازه‌های زیر با توجه به اندازه‌های تصاویر name@2x.png ساخته شده که برای صفحه‌های بزرگتر ساخته شده‌اند. اما شما می‌توانید این اندازه‌ها را به شکلی که نسبت‌ها و حس کلی بازی حفظ شود تغییر دهید.

ارتفاع	عرض	موجودیت
۱۰۲۴	۶۴۰	صفحه‌ی بازی (پس‌زمینه)
۱۸۴	۱۳۶	دشمن (پرنده‌ی سه‌چشم)
۳۶	۱۲۰	سکو (پلتفرم)
۱۲۰	۱۲۴	کاراکتر بازی
۴۶	۵۶	فنر

نکات تکمیلی

- در این تمرین تعداد زیادی پارامتر وجود دارد (مانند سرعت حرکت افقی و جاذبه‌ی زمین و...) از شما انتظار می رود که طبق صلاح‌دید خودتان این اعداد را تنظیم کنید به صورتی که حتی الامکان بازی مانند بازی داده شده در لینک شود (خیلی روی اعداد دقیق حساس نباشید).
- فایل دنباله‌ها با فرض اندازه‌ی صفحه‌ی بازی برابر با ۱۰۲۴×۶۴۰ است. اگر این اندازه زیادی برای صفحه‌ی کامپیوتر شما بزرگ است می‌توانید اندازه‌ی صفحه و اجسام بازی را به یک نسبت کوچک کنید و لازم است که مکان سکوها و غیره که از فایل خوانده می‌شود را نیز با همین نسبت کوچک کنید تا در صفحه جا شوند و بازی قابل اجرا باشد.

نحوهٔ تحویل

- تمام فایل‌های خود را در قالب یک **پرونده‌ی zip** با نام A5-<SID>.zip در صفحهٔ ایلرن درس بارگذاری کنید که SID شمارهٔ دانشجویی شماست؛ برای مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۹۹۹۹۹ است، نام پروندهٔ شما باید A5-810199999.zip باشد.

- **دقت کنید** که پرونده‌ی zip آپلودی شما باید پس از Unzip شدن شامل پرونده‌های پروژه شما (از جمله Makefile) باشد و از zip کردن پوشه‌ای که داخل آن فایل‌های پروژه‌تان قرار دارد خودداری فرمایید. برای مثال، نمونه فایل مورد قبول در ادامه آمده است:

```
A5-810199999.zip
├── main.cpp
├── makefile
└── ...
```

- برای ایجاد رابط کاربری گرافیکی[5] و تمامی افکت‌های برنامه خود باید از کتابخانه‌های SDL2 و RSDL استفاده کنید.

- فایل بارگذاری شده توسط شما باید پوشه‌ی کامل پروژه باشد که شامل کد کامل برنامه شما به همراه کتابخانه‌ی RSDL، تصاویر و سایر موارد است.

[5] GUI

- در این تمرین بازی شما توسط دستیاران آموزشی آزموده می‌شود و تست خودکار ندارد.
- نمره هر بخش در صورت صحت عملکرد آن در بازی شما اختصاص می‌یابد و داشتن کد یک بخش که در بازی قابل آزمودن نیست نمره‌ای برای شما **ندارد**.
- برنامه شما باید حتماً طراحی شیءگرا داشته باشد.
- **دقت کنید** که پروژه‌ی شما باید Multi-file باشد و Makefile داشته باشد. همین‌طور در Makefile خود مشخص کنید که از استاندارد c++11 استفاده می‌کنید.
- دقت کنید که نام پرونده‌ی اجرایی شما باید doodleJump.out باشد.
- طراحی درست، رعایت سبک برنامه‌نویسی درست و تمیز بودن کد برنامه‌ی شما در نمره‌ی تمرین تأثیر زیادی دارد.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.