دانشکده‌ی مهندسی کامپیوتر

پروژه‌ی طراحی سیستم‌های دیجیتال

مدرس:

دکتر فلاحتی

دستیاران آموزشی:

صبا مستوفی

سامان محسنی

مجید طاهرخانی

فاطمه خاشعی

محمدعلی پشنج

نکات کلی:

- پروژه درس طراحی سیستم‌های دیجیتال در **۳** فاز انجام می‌گیرد. در فاز اول، دانشجویان گرامی باید تیم خود و پروژه‌ای که قرار است روی آن کار کنند را معرفی نمایند. در فاز دوم، لازم است تا طرح اولیه و چارت‌های FSM و ASM را برای آن ترسیم نموده و در آن عملیات ماژول‌های مختلف مشخص شود (در صورت نیاز برای اجزا چارت‌های مستقل رسم کنید). در فاز سوم نیز انتظار می‌رود طراحی نهایی بخش‌های مختلف انجام گیرد.
- گروه‌ها دو نفره (در شرایط خاص ۳ نفره) هستند.
- علاوه بر موضوعات پیشنهادی، دانشجویان گرامی می‌توانند پروژه‌های پیشنهادی خود را ارائه کنند و در صورت تایید، کار خود را بر روی آن پروژه شروع کنند.
- در هر پروژه، ماژول‌های لازم و خروجی مورد انتظار تعریف شده‌اند. در جزییات طراحی می توانید با ابتکار خودتان تغییراتی ایجاد کنید و یا حتی بخش‌های خلاقانه‌ای به پروژه‌ها اضافه کنید و با توجه به میزان و کیفیت بخش‌های اضافه شده، نمره بیشتری دریافت کنید.
- **توجه کنید که در تمامی بخش‌های این پروژه، ماژول‌ها باید سنتزپذیر باشند.**
- توصیه اکید می‌شود انجام پروژه را به تدریج از زمان تعریف آن آغاز کنید و هر فاز را در زمان مشخص شده تحویل دهید.

در ادامه لیستی از موضوعاتی پیشنهادی برای پروژه درس طراحی سیستم دیجیتال قرار داده شده است.

پروژه‌های پیشنهادی

۱. آسانسور (سطح ساده- امتیاز ۴۰)

دو آسانسور برای یک ساختمان ۵ طبقه‌ای طراحی کنید که حداکثر وزن قابل تحمل هر یک از آن‌ها ۵۰۰ کیلوگرم (۶ نفر) باشد. شما باید قابلیت نشان دادن اینکه آسانسور در چه طبقه‌ای قرار دارد و به چه سمتی در حال حرکت است را داشته باشید. اگر وزن افراد حاضر در آسانسور از حداکثر وزن قابل تحمل بیشتر باشد، آسانسور نباید حرکت کند. اگر هر دو آسانسور در حال حرکت به یک سمت بودند (مثلا به سمت بالا) در صورت فشردن دکمه آسانسور، آسانسوری که تعداد افراد کمتری دارد در آن طبقه بایستد. برای ورود هرشخص به آسانسور یک دکمه تعبیه شود که با فشردن آن به تعداد افراد داخل آسانسور یک نفر اضافه شود و یک دکمه برای خروج هر شخص نیز وجود داشته باشد. همچنین توسط یک سون سگمنت طبقه‌ای که آسانسور در آن قرار دارد و تعداد افراد داخل آسانسور نشان داده شود.

نکته ۱: در طراحی الگوریتم‌های حرکتی آسانسور می‌توانید اهداف متفاوتی را دنبال کنید، مانند کم‌ترین زمان از دید کاربر، کم‌ترین مسافت طی‌شده آسانسور، حرکت در یک راستا. هدف مورد نظر خود و دلیل آن را بیان کنید و در تعیین جهت حرکت آسانسور از آن استفاده نمایید.
نکته ۲: شما می‌توانید موارد دیگری مانند تماس‌های اضطراری، تعیین محل توقف آسانسور، تغییر الگوریتم‌ها بر اساس وضعیت استفاده از آسانسور را در نظر بگیرید.
نکته ۳: با توجه به کیفیت و کارایی الگوریتم‌های حرکتی و امنیتی، پروژه در سطح بالاتر (متوسط و پیشرفته) در نظر گرفته می‌شود.

۲. طراحی چراغ راهنمایی هوشمند (سطح متوسط- امتیاز ۵۰)

یک چراغ راهنمایی برای یک چهارراه با یک خیابان اصلی و سه خیابان فرعی طراحی کنید. این چراغ راهنمایی دارای ۳ حالت مختلف کاری است.

حالت اول (حالت عادی): در این حالت مدت زمان سبز بودن چراغ خیابان اصلی، اندکی بیشتر از خیابانهای فرعی است. مگر اینکه حجم ترافیک در خیابان های فرعی خیلی بالا باشد (می‌توان با فشردن دکمه‌ای، ورود ماشین به خیابان را نشان داد و تعداد آن‌ها را ذخیره کرد و اگر بیشتر از یک تعداد مشخص بود، حجم ترافیک را بالا در نظر گرفت). در این حالت زمان سبز بودن چراغ خیابان‌های فرعی اندکی افزایش می‌یابد تا از حجم ترافیک آن‌ها کاسته شود.

حالت دوم (حالت ازدحام): ساعتی از شبانه روز، حجم ترافیک روی خیابان اصلی بسیار بالا است و باعث افزایش ترافیک سایر مسیرها و تقاطع‌ها نیز می‌شود. در این حالت، زمان سبز بودن چراغ راه اصلی، بسیار بیشتر از راه‌های فرعی است.

حالت سوم (حالت چشمک زن): در نیمه شب که عبور و مرور زیادی در این تقاطع نیست، چراغ به حالت سوم می‌رود و به صورت چشمک زن در می‌آید.

۳. آبیاری گیاهان هوشمند (سطح متوسط- امتیاز ۵۰)

یک سیستم طراحی کنید که با توجه به شرایط محیطی (نور، دما و ...) و همچنین میزان رطوبت خاک اقدام به آبیاری گیاهان کند. آبیاری تا زمانی ادامه دارد که میزان رطوبت خاک از یک مقدار مشخص شده بیشتر شود. برای میزان آبیاری گیاهان محدودیت وجود دارد و در شبانه روز بیشتر از ۱ ساعت نمی‌توان گیاهان را آبیاری کرد. شما باید قابلیت نشان‌دادن میزان رطوبت خاک را در هر لحظه داشته باشید. اگر به محدودیت آبیاری گیاهان رسیده بودیم و رطوبت خاک از یک مقدار مشخص‌شده کمتر باشد، اخطار نشان داده شود. شرایط محیطی مانند نور، دما و ... به صورت یک عدد در ورودی‌ها در دسترس است و میزان رطوبت هوا توسط یک سون سگمت نشان داده شود.

٤. خانه هوشمند (سطح متوسط- امتیاز ۵۰)

سیستمی طراحی کنید که برای یک اتاق در خانه موارد زیر را کنترل کند:

- در هر لحظه تعداد افراد داخل اتاق را نشان دهد (برای مثال، با فشردن یک دکمه، یک فرد به افراد داخل اتاق اضافه می‌شود).
- دمای محیط اتاق را نشان دهد. اگر دما از ۲۸ درجه سانتیگراد بیشتر شد، کولر را روشن کند. در صورتی‌که دمای اتاق از ۲۵ درجه سانتیگراد کمتر بود، کولر خاموش شود. برای نشان دادن روشن و یا خاموش بودن کولر می‌توانید از یک LED استفاده کنید.
- نور اتاق تشخیص داده شود و در صورت کمتر بودن نور از مقدار مشخصی، لامپ اتاق روشن شود. نور اتاق به صورت یک عدد ورودی در دسترس است.
- اگر در اتاق دود مشاهده شد تشخیص دهد و اخطار نشان داده شود.

۵. بازی Tic Toc Toe (سطح متوسط – امتیاز ۵۰)

در این بازی، دو بازیکن با هم به رقابت می‌پردازند. هر بازیکن می‌تواند خانه مورد نظرش را با فشردن دکمه‌ای از روی صفحه کلید انتخاب کند. سپس آن خانه‌ای که بازیکن انتخاب کرده است، با چراغی با

رنگ متناظر به آن بازیکن روشن می‌شود. برای برنده شدن، مشابه بازی اصلی، کافی است خانه های پشت هم (افقی، عمودی، ضربدری) توسط یک بازیکن روشن شود. نکاتی که باید مد نظر قرار گیرد:

۱ - حرکت غیرمجاز نتواند انجام شود. یعنی خانه‌ای که توسط یک بازیکن انتخاب می‌شود، قبلا توسط بازیکن دیگری انتخاب نشده باشد.

۲ -اگر همه خانه‌ها پر بود ولی برنده‌ای وجود نداشت، بازی مساوی اعلام شود.

۳- نمایش خروجی می‌تواند به صورت ماتریس LED در رنگ‌های مختلف باشد.

۶. سامانه‌ی پارکینگ هوشمند (سطح پیشرفته - امتیاز ۸۰)

یک سامانه هوشمند برای مدیریت پارکینگ طراحی کنید. این سامانه اجازه ورود خودرو، محل پارک، تخصیص توکن، خروج خودرو، محاسبه ظرفیت پارکینگ و محاسبه زمان پارک خودرو را مشخص می‌کند و اطلاعات مربوط به هر بخش را در خروجی نشان می‌دهد. در این سامانه اطلاعات ورودی یعنی ظرفیت پارکینگ و الگوی ورودی مربوط به توکن قابل تنظیم است. سامانه مورد نظر شامل چندین ماژول است:

ماژول ۱: مدار ورود خودرو (تشخیص ورود خودرو و تخصیص محل پارک)

این ماژول ورود خودرو به پارکینگ را با استفاده از سنسورهای مادون قرمز شناسایی می‌کند. در صورت وجود ظرفیت، ماشین وارد پارکینگ می‌شود. پس از ورود، مدار مربوط به تخصیص شماره محل پارک فعال می‌شود. به این ترتیب، ماشین به اولین/نزدیک‌ترین فضای پارک خالی موجود هدایت می‌شود (می‌توانید از انکدر برای طراحی این ماژول استفاده کنید).

- ظرفیت قرارگیری خودروها در پارکینگ به صورت یک ورودی ۸ بیتی به سامانه داده می‌شود. در صورت ورود یا خروج خودرو ظرفیت جایگاه پارک بروزرسانی می‌شود.
- ظرفیت هر جای خالی پارک با بیت ۱ مشخص می‌گردد. در صورت صفر شدن، در آن محل خودرو پارک شده است.

ماژول ۲: مدار تولید توکن (مدار تخصیص توکن به خودرو)

بعد از مشخص شدن محل پارک، به خودرو توکنی تخصیص داده می‌شود. توکن اختصاص داده شده برای خروج از پارکینگ، محاسبه زمان پارک، پرداخت فاکتور و ذخیره اطلاعات مربوط به خودرو مورد استفاده قرار می‌گیرد. توکن با استفاده از شماره پارک و یک الگو تولید می‌شود. برای این کار، شماره پلاک و الگو را با هم xor (bitwise) کنید.

ماژول ۳: مدار خروج خودرو (مدار خروج از پارکینگ)

ماژول خروج خودرو، خروج خودرو از محل را توسط سنسورهای مادون قرمز شناسایی می‌کند. این ماژول به کمک توکن تخصیص داده شده به هر خودرو، محل پارک را شناسایی، هزینه پارک و ظرفیت پارکینگ را محاسبه می‌کند. برای این منظور، کاربر باید الگوی ورودی (در زمان ورود به ماشین اختصاص داده شده است) و توکن را ارائه دهد. در ادامه، ماژول خروجی سه عملیات انجام می‌دهد: (۱) توکن ثبت شده در سیستم و توکن ورودی کاربر را تطبیق می‌دهد و در صورت مطابقت، به خودرو اجازه خروج داده می‌شود. (۲) با استفاده از الگوری ورودی و توکن، محل پارک خودرو و هزینه پارک را مشخص می‌کند. (۳) ظرفیت پارکینگ و و ضعیت جایگاه‌های پارک را به روز می‌کند. دقت کنید، ظرفیت پارکینگ را باید نمایش دهید.

ماژول ۴: مدار محاسبه کننده‌ی زمان (مدار محاسبه کننده زمان حضور خودرو در پارکینگ)

ماژول زمان، زمان را بر اساس زمان ورود و زمان خروج خودرو محاسبه می‌کند. زمان ورود و خروج خودرو بر اساس سنسورهای مادون قرمز ثبت می‌شود.

```
time = timeout - timein
```

ماژول ۵: مدار شمارنده ظرفیت پارکینگ (اختیاری)

با استفاده از ظرفیت محاسبه شده در ماژول ۳، مداری طراحی نمایید که تعداد یک و صفر را شمرده و مقدار فضای خالی و پارک شده را مشخص نماید.

الف) مدار شکل ۶ یک حالت انتزاعی از طرح را نشان میدهد. پس از طراحی توابع مورد نیاز، آن را با استفاده از زبان وریلاگ توصیف نمایید. در طراحی خود میتوانید از هر بالک منطقی مانند گیت‌های پایه، جمع کننده ـ تفریق کننده، مقایسه کننده، دیکدر، انکدر و مالتی پلکسر استفاده نمایید. توجه داشته باشید که توصیف شما باید به صورت ساختاری باشد.

۷. طراحی یک واحد پردازشی ضرب دو ماتریس (سطح پیشرفته-امتیاز ۱۰۰)

یک واحد پردازشی ضرب دو ماتریس، با استفاده از الگوریتم تقسیم و حل (divide and conquare) طراحی کنید. این الگوریتم موازی‌سازی عملیات را به نحو مطلوبی فراهم می‌کند.

مؤلفه‌های ماتریس از نوع اعشاری با دقت یگانه (Point Floating Precision Single) هستند. فرمت اعداد اعشاری مطابق با استاندارد IEEE754 است. یک واحد ضرب اعشاری به همراه برنامه تست ، هر دو به زبان وریلاگ در اختیار طراحان قرار داده می‌شود. بنابراین طراحی واحد پایه‌ی ضرب‌کننده اعشاری در دستور کار این پروژه نیست بلکه گسترش این ضرب‌کننده برای انجام عملیات ضرب ماتریس

مد نظر است. به گونه ای که موازات خوبی در ضرب ماتریس‌ها داشه باشیم (راهنمایی: موازات در اجرای هر یک از درایه‌ها).

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{21} & X_{22} & X_{23} & X_{24} \\ X_{31} & X_{32} & X_{33} & X_{34} \\ X_{41} & X_{42} & X_{43} & X_{44} \end{bmatrix} \times \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} & Y_{14} \\ Y_{21} & Y_{22} & Y_{23} & Y_{24} \\ Y_{31} & Y_{32} & Y_{33} & Y_{34} \\ Y_{41} & Y_{42} & Y_{43} & Y_{44} \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} & Z_{13} & Z_{14} \\ Z_{21} & Z_{22} & Z_{23} & Z_{24} \\ Z_{31} & Z_{32} & Z_{33} & Z_{34} \\ Z_{41} & Z_{42} & Z_{43} & Z_{44} \end{bmatrix}, \quad (1)$$

where

$$Z_{11} = X_{11}Y_{11} + X_{12}Y_{21} + X_{13}Y_{31} + X_{14}Y_{41}, \quad (2)$$

$$Z_{12} = X_{11}Y_{12} + X_{12}Y_{22} + X_{13}Y_{32} + X_{14}Y_{42}, \quad (3)$$

and so on...

- عملیات ضرب ماتریس در چند سیکل انجام می‌شود و از مدار ترتیبی (sequential) و ترکیبی (combinational) است. بهین سازی تعداد سیکل‌ها در مقایسه با مساحت سخت‌افزار یک کار موازنه‌ای است که باید حالت بهینه را از دیدگاه خودتان گزارش کنید. به عبارت دیگر، علاوه بر سنتزپذیر بودن، تولید نتیجه بهینه از نظر مساحت و زمان اهمیت دارد.
- بهینه‌سازی مساحتی و سبک کدینگ واحد ضرب‌کننده اعشاری که در اختیارتان گذاشته می شود نیز به کیفیت کار شما و در نتیجه، نمره ارزیابی پروژه کمک خواهد کرد.
- ابعاد ماتریس‌ها به صورت پارامتر در کد خواهد بود، به این صورت که با تغییر این پارامتر، ماتریس‌هایی با ابعاد مختلف می‌توانند در هم ضرب شوند. محتویات ماتریس‌های ورودی از یک (یا دو فایل) دریافت می‌شود. برای انجام این کار می‌توانید در برنامه تست، بسته به پارامتر اندازه ماتریس‌ها، دو فایل محتوی اعداد تصادفی اعشاری تولید کنید.

۸. شبیه‌سازی حرکت یک طناب با روش: Verlet integration (سطح پیشرفته- امتیاز ۱۲۰)

در این پروژه قرار است با کمک FPGA حرکت یک طناب را شبیه‌سازی کنید. طناب را می‌توان مجموعه‌ای از گره‌های متصل به هم فرض کرد و سپس با اعمال فیزیک بین گره‌ها، شبیه‌سازی را انجام داد. برای انجام این کار می‌توان از روش Verlet integration یا هر روش دیگری که دوست دارید استفاده کرد. در آدرس زیر کدی به زبان پایتون برای نمایش کلیت چیزی که از شما انتظار داریم قرار داده شده است و در مرورگر قابل اجرا و تغییر است (برای باز کردن لینک نیاز به VPN دارید) .

https://trinket.io/python/8720f8dd81?runOption=run

با نزدیک‌کردن نشانگر mouse به گره‌ها، می‌توان به گره‌ها ضربه زد. در این الگوریتم در تابع update_dots() ابتدا موقعیت هر گره به صورت مستقل بر اساس موقعیت فعلی (dots[i].x, dots[i].y) و موقعیت پیشین (dots[i].px, dots[i].py) و جاذبه update شده و سپس موقعیت گره‌ها نسبت به هم تنظیم می‌شوند، به این صورت که ابتدا موقعیت گره با اعمال محدودیت فاصله با گره بالایی بدست آمده (nxu, nyu) و سپس موقعیت گره با اعمال محدودیت فاصله با گره پایینی بدست می‌آید (nxd, nyd) و میانگین این دو به عنوان موقعیت نهایی گره در این لحظه در نظر گرفته می‌شود. برای پیاده‌سازی از تعدادی واحد پردازشی که به صورت زنجیروار با یکدیگر ارتباط دارند استفاده کنید. تعداد گره‌ها می‌تواند بیشتر از تعداد واحدهای پردازشی باشد، بنابراین باید بتوان چند گره را درون واحدها قرار داد. همچنین با کمک روش‌های خلاقانه و تخمینی (در صورت نیاز) واحدها را ساده‌تر کنید. (به عنوان مثال حتما نیازی به استفاده از تقسیم‌کننده نیست و می‌توان از روش Fast inverse square root برای محاسبه (nxu, nyu) و (nxd, nyd) کمک گرفت). در نهایت شکل حاصل را درون یک بافر گرافیکی ترسیم و با کمک یک کنترلر VGA روی صفحه نمایش دهید.

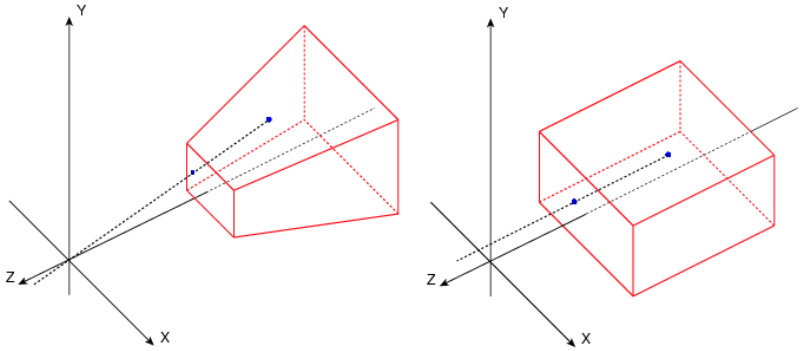
۹. پیاده‌سازی کنسول بازی به همراه یک بازی ساده (سطح پیشرفته- امتیاز ۱۴۰)

در این پروژه قرار است یک کنسول بازی ساده طراحی و پیاده‌سازی کنید، سپس یک بازی بسیار ساده به دلخواه خودتان برای اجرا بر روی این کنسول طراحی و کد نویسی کنید.

مقدمه:

در بازی‌های گرافیک سه بعدی اطلاعات مربوط به اجسام معمولا به صورت مجموعه vertex هایی در فضا می‌باشند (Triangle mesh) و تحرک اجسام از طریق جابه‌جایی همین vertex ها انجام می‌شود. در ساده‌ترین حالت، برای نمایش محیط بازی روی صفحه، vertex ها بر روی صفحه‌ای مجازی تصویر می‌شوند. از اتصال vertex ها، مثلث‌هایی تشکیل می‌شوند که رنگ هر پیکسل را مشخص می‌کنند.

نگاشت vertex ها روی صفحه در بازی‌های سه بعدی معمولا از نوع perspective (شکل ۱ سمت چپ) و در بازی‌های دو بعدی معمولا از نوع orthographic (شکل ۱ سمت راست) می‌باشد.



شکل ۱

جزئیات پیاده‌سازی:

این کنسول شامل یک پردازنده مرکزی است؛ برای پیاده‌سازی این واحد می‌توانید از هسته‌های soft microprocessor آماده یا پیاده‌سازی‌های پردازنده MIPS ساده موجود استفاده کنید و یا خودتان یک پردازنده MIPS بنویسید.

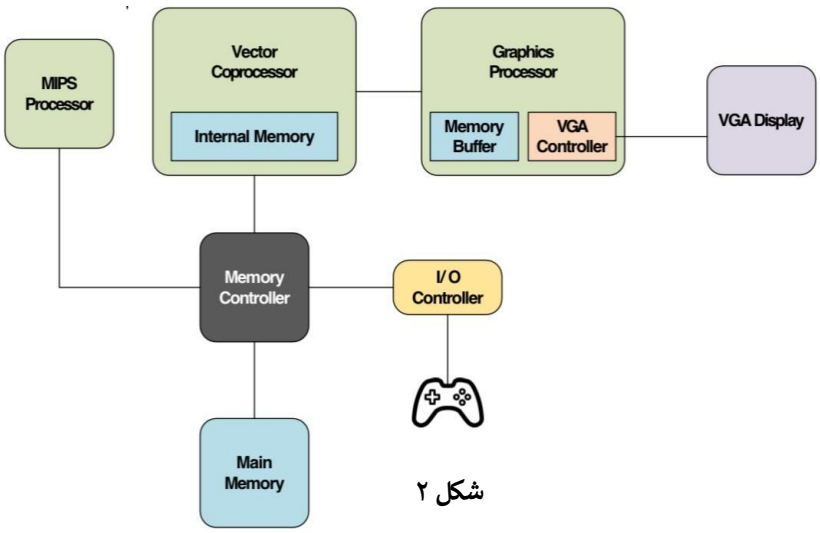
همچنین این کنسول داری یک واحد vector coprocessor، برای انجام محاسبات ساده گرافیکی مثل انتقال vertex ها می‌باشد. براساس ویژگی‌های بازی، توانایی محاسباتی این واحد تعیین می‌شود. به عنوان مثال برای یک بازی دو بعدی بسیار ساده، احتمالا واحدهای جمع‌کننده و ضرب‌کننده برداری برای اعداد صحیح کافی می‌باشد. بازی‌های سه بعدی نیازمند واحدهای محاسبات برداری اعشاری (شامل تقسیم کننده برداری اعشاری) می‌باشد.

یک واحد پردازنده گرافیکی بسیار ساده نیز در این کنسول موجود است که وظیفه رنگ‌آمیزی مثلث‌ها را بر عهده داشته و حاصل را به بافر گرافیکی می‌فرستد. در ادامه، یک کنترلر VGA مقدار بافر حافظه گرافیکی را روی صفحه، نمایش می‌دهد.

معماری شکل ۲ مثالی از نحوه ارتباط بخش‌های مختلف سیستم را نشان می‌دهد. در این معماری واحدهای مختلف از طریق تقسیم فضای آدرس‌دهی حافظه با یکدیگر در ارتباط هستند. کنترل‌کننده حافظه با دانستن فضای آدرس‌دهی هر واحد، داده‌ها را به مقصد مورد نظر ارسال می‌کند. کد بازی شامل دو بخش کد پردازنده MIPS و کد vector coprocessor می‌باشد.

کد MIPS منطق بازی و کنترل بخش‌های مختلف را به عهده دارد. کدهای مربوط به vector coprocessor در بخشی از حافظه مربوط به این واحد کپی کرده و با نوشتن در رجیسترهای کنترلی آن مشخص می‌کند چه محاسباتی روی کدام داده‌ها انجام شود. سپس vertex های نهایی از vector

coprocessor به graphics processor ارسال شده و از روی آن مقادیر پیکسل‌ها مشخص شده و در memory buffer قرار می‌گیرد.



شکل ۲

معماری و روش‌های ارائه شده صرفا جهت ایده دادن مطرح شده و شما در صورت تمایل می‌توانید تغییراتی به دلخواه خود اعمال کنید یا از روش دیگری استفاده کنید. همچنین توصیه می‌شود برای جلوگیری از پیچیدگی بیش از حد، بازی دو بعدی و ساده‌ای مانند شکل ۳ را پیاده‌سازی کنید. برای اجرای این بازی vector coprocessor تنها نیاز به یک جمع‌کننده برداری برای انتقال اجسام دارد.

شکل 3

موفق باشید