به نام خدا

"منچ"

مستندات بازی

استاد : خانم فاطمه صفی نیا

دانشجو : سید محمد میررجبی

بهمن ماه 1394

با سلام.

با توجه به واضح بودن شرایط و قوانین بازی و مانند آن، سند جمع آوری شده صرفا مدرکیست جهت توضیح روش انجام پروژه و در آن به توضیح قوانین و اصول بازی بطور مجزا پرداخته نمیشود.

لینک های مراجعه برای کسب اطلاعات بیشتر :

https://en.wikipedia.org/wiki/Ludo_%28board_game%29

http://www.mastersgames.com/rules/ludo-uckers-rules.htm

نمایی از اجرای برنامه

جزئیات کد و روش کار :

چرخه ی کلی بازی به شکل زیر است :

```
bool _finished = false;
int main(){
        //آماده سازی های اولیه
        while(!_finished){
                Update();
        }
        //پایان بازی
}
void Update()
{
    RepaintMap();//رسم صفحه بازی
    RollDice();//انداختن تاس و بررسی آن
    Log();//چاپ جزئیات و رویداد های بازی
}
```

به این حالت که در هر مرحله شرط پایان بازی بررسی میشود(رسیدن هر **4** مهره ی یکی از بازیکنان به خانه های پایانی) و با توجه به آن حلقه تکرار یا خاتمه پیدا میکند. همچنین در هر مرحله ابتدا صفحه ی بازی پس از اعمال آخرین تغییرات رسم میشود و سپس با توجه به نوبت بازیکن فعلی تاس انداخته شده و برای همان بازیکن پردازش های لازم صورت میگیرد

.

کلاس ها :

1.کلاس **Piece** (مهره) : با در نظر گرفتن اینکه در بازی هر بازیکن دارای **4** مهره است و به ازای هر مهره حالاتی خاص وجود دارد. برای هر مهره کلاسی در نظر گرفته ام که دارای مشخصات زیر است و متناسب با هر کدام تابعی برای تغییر آنها (به سبک و سیاق **Property** های دات نت) در نظر گرفته شده

- خصوصی :
  - شناسه مهره (**id**_) : که جهت سهولت در شناسایی و متمایز کردن مهره ها در نظر گرفته شده.
  - موقعیت مکانی مهره (**position**_): که مکان مهره را بر روی صفحه در خود ذخیره میکند.
  - شمایل مهره (**face**_) : صرفا کاراکتری است برای متمایز نمودن مهره ها از هم.
- عمومی :
  - تعداد خانه های گذرانده(**Passed**) : هر مهره در خارج از زمین مقدار **1-** را در این متغییر خود دارد و همچنین هر مهره در داخل زمین مقداری بین **0** تا **39** در این متغییر خود دارد و در حالتی که این متغیر **40** یا بیشتر شود گوییم مهره از زمین خارج شده و آنرا در خانه های متناسب تا پایان بازی رسم میکنیم.
  - مکان مهره در آرایه ی مسیر(**Step**) : اندیسی است که مشخص میکند مهره در کدام نقطه از مسیر قرار دارد. از این متغییر برای بررسی برخورد ها و حرکت کردن مهره ها به ساده ترین روش بر روی مسیر استفاده شده.

2. کلاس **Player** (بازیکن) : در بازی دو بازیکن داریم که ساختاری اینچنینی دارند :

- خصوصی :
  - نام (**name_**) : که جهت ایجاد تمایز و واضح تر شدن روند بازی است.
- عمومی :
  - تعداد مهره های در حال بازی که در زمین هستند (**PiecesInsideCount**)
  - تعداد مهره های خارج از بازی که وارد زمین نشده اند یا توسط بازیکن حریف به بیرون زده شدند (**PiecesOutsideCount**)
  - تعداد مهره هایی که یک دور کامل زده اند و در خانه های پایانی قرار دارند (**PiecesInHouseCount**)
  - آرایه ی مهره های داخل (**PiecesInside**)
  - آرایه ی مهره های خارج (**PiecesOutside**)
  - آرایه ی مهره های به پایان رسانیده (**PiecesInHouse**)
  - عدد مشخص کننده ی حریف (**Opponent**)

(در زمان آغاز پروژه کار را با آرایه های معمولی شروع کردم و در نقطه ای به این نتیجه رسیدم که استفاده از **vector** ساده تر بود که دیگر تقریبا توابع مورد نیازی که **vector** در اختیارمان قرار میدهد را نوشته بودم و وقت لازم جهت انجام تغییرات در اختیار نبود.)

توابع کلاس **Player** :

دسته ای از این توابع صرفا جهت مرتب سازی و تغییرات ساده در آرایه هاست :

- **ReArrangePieces()** : جهت مرتب کردن آرایه ی مهره های داخل زمین به اولویت بزرگتر بودن مقدار **Passed** در هرکدام.
- **AddInsidePiece(Piece piece)** : افزودن یک مهره به آرایه مهره های داخل.
- **RemoveInsidePieceAtPosition(int index)** : حذف یک مهره از آرایه مهره های داخل در نقطه ی مورد نظر.
- **RemoveInsidePieceById(int id)** : حذف یک مهره از آرایه مهره های داخل با شناسه.
- **AddOutsidePiece(Piece piece)** : افزودن یک مهره به آرایه مهره های بیرون.
- **RemoveOutsidePieceAtPosition(int index)** : حذف یک مهره از آرایه مهره های بیرون در نقطه ی مورد نظر.
- **&GetInsidePieceById(int id)** : گرفتن ارجاعی مهره ای در داخل زمین برای اعمال تغییرات.
- **&GetOutsidePieceById(int id)** : گرفتن ارجاعی مهره ای در بیرون زمین برای اعمال تغییرات.

دسته بعد توابع کنترل کننده ی جریان بازی اند :

- **InitiatePieces()** : مقدار دهی مهره ها.
- **PrintOptions(int dice)** : چاپ انتخاب های کاربر پس از انداختن تاس.
- **ClearOptions()** : پاک کردن بخش انتخاب های کاربر.
- **HandleDice(int dice)** : دریافت تاس و اعمال حرکت با توجه به مقدار تاس.
- **MoveToHouse(int id)** : بردن یک مهره از داخل که بیش از **40** خانه را گذرانده به خانه های پایانی.
- **BringPieceIn()** : آوردن یک مهره از خارج به داخل زمین.
- **MovePiece(int id,int step)** : حرکت دادن یک مهره با توجه به تاس آورده.
- **DropOut(int id, char hit)** : انداختن مهره ی بازیکن به بیرون زمین.

نکات :

- برای تست بازی بدلیل فراهم آوردن شرایط آزمایشگاهی در تابع **RollDice** سه خط کامنت شده اند که از آنها میتوان برای ایجاد تاس دلخواه و بررسی نتیجه استفاده کرد.
- در بازی یک سیستم ثبت وقایع(**Log**) تعبیه شده که تک تک رویداد های بازی را در فایل **log.txt** در همان پوشه ذخیره میکند که در هر بازی از ابتدا نوشته میشود.
- از کتابخانه ی **rlutil**[1] جهت اعمال رنگ بندی ها استفاده شده.

با آرزوی توفیق

سید محمد میررجبی

http://Mirrajabi.ir

mohammad@mirrajabi.ir

mohammadmirrajabi@gmail.com

[1] http://tapiov.net/rlutil/