**« به نام او »**

# آموزش شبیه سازی دو بعدی فوتبال

نویسندگان : محمدعلی میرزایی - علی یعقوبی

**مرجع روبوکاپ ایران**

**www.iranrcss.com**

**مباحث این جلسه :**

- آموزش ساختار و کدنویسی در بیس UVA_Trilearn
- مباحث مربوط به ارث بری، ایجاد و استفاده کتابخانه ها و پارامتر های ایجنت

مقدمه :

برای شروع شبیه سازی و ساخت تیم دو راه وجود دارد. راه حل اول این است که خودتان از ابتدا با مطالعه manual سرور و پارامتر های مورد نیاز برای ایجاد کانکشن و ... یک تیم از ابتدا پایه گذاری کنید و راه حل دوم که برای شروع کار توصیه میشود این است که از بیس های موجود که توسط سایر افراد نوشته شده است استفاده نمایید. برای انجام راه حل اول نیاز به تجربه و داشتن اطلاعات گسترده در زمینه برنامه نویسی و شبکه دارید و معمولا به افرادی که تازه، وارد این رشته شده اند، بیس نویسی توصیه نمیگردد. در متد آموزشی مرجع روبوکاپ ایران، آموزش ها از ابتدا و کامل بر روی بیس ها ارائه میشود.

برای سهولت کار و یادگیری بیشتر، در بدو کار، آموزش بر روی بیس UVA_Trilearn آغاز میگردد. با ادامه آموزش ها و رسیدن به سطح مطلوب، مقاله های آموزشی سایر بیس ها و همچنین مقالات مربوط به هوش مصنوعی ( Artificial Intelligence ) در دستور کار قرار میگیرد.

شروع با UVA_Trilearn :

برای شروع کار با بیس یو وی ای، با کمک آموزش های قسمت سوم، بیس را نصب نمایید و تغییرات مورد نظرتان ( از قبیل تغییر نام تیم و ... ) را انجام دهید. سپس به پوشه src بروید. در این پوشه کلیه فایل های مورد نیاز برای کد نویسی تعریف شده است.

ابتدا وارد پوشه src شده و با استفاده از یکی از editor های موجود ، فایل main.cpp را باز می کنیم .

تابع main() :

ما در ابتدا به شناخت این تابع احتیاج نداریم ، اما دلیلی که باعث شد تا این تابع در این مقطع معرفی شود ، وجود فراخوان تابع bp.mainLoop() می باشد . تابع mainLoop تابعی است که در آن تصمیم گیری های لازم برای بازی گرفته می شود . در ادامه به بررسی این تابع که در فایل player.cpp می باشد ، می پردازیم .

تابع mainLoop() :

این تابع اصلی ترین تابع تصمیم گیری بیس است . تابعی که در آن براساس playertype بازیکن به یکی از توابع تصمیم گیری زیر فراخوانی می شود :

```
goalieMainLoop( )
defenderMainLoop( )
midfielderMainLoop()
attackerMainLoop()
```

- در مورد player types ها بعدا توضیح خواهیم داد .

حال به بررسی هر کدام از توابع بالا می پردازیم .

تابع goalieMainLoop( ) :

در این تابع تابع دیگری به نام deMeer۵_goalie() فراخوانی می شود که این تابع در فایلplayerTeams.cpp می باشد.

تابع defenderMainLoop( ) :

در این تابع تابع دیگری به نام deMeer۵() فراخوانی می شود که این تابع در فایل playerTeams.cpp می باشد.

تابع midfielderMainLoop( ) :

در این تابع تابع دیگری به نام deMeer۵() فراخوانی می شود که این تابع در فایل playerTeams.cpp می باشد.

تابع attackerMainLoop( ) :

در این تابع تابع دیگری به نام deMeer۵() فراخوانی می شود که این تابع در فایل playerTeams.cpp می باشد.

تابع deMeer۵() :

ورود شما را به دنیای روبوکاپ تبریک می گویم . بله از اینجا کار یک برنامه نویس روبوکاپی آغاز می شود. جایی که تمام دستورات لازم به بازیکن داده می شود . این تابع در هر سیکل برای هر ۱۰ بازیکن اجرا می شود . حال این شمایید که به بازیکن بگویید که در هر سیکل چه کاری انجام دهد . در ابتدای این تابع متغییری از نوع soccerCommand تعریف شده است . این متغییر ، کاری است که در انتها شما به سرور می فرستید تا بازیکن به آن عمل کند . مثلا شوت یا پاس یا ... هر کدام یک soccerCommand می باشد . در خط بعدی متغییری به نام posAgent از نوع VecPosition تعریف شده است . VecPosition یک بردار است که ابتدای آن مرکز زمین و انتهای آن معرف نقط مورد نظر است . به طور مثال نقطه ی (۲۰,-۱۰) VecPosition a نقطه ایست که ابتدای آن نقطه (۰,۰) و انتهای آن نقطه ی (۲۰,-۱۰) می باشد . posAgent همانطور که از اسمش پیداست مختصات مکان بازیکنی است که در حال حاضر تابع برای او اجرا می شود . در خط بعدی نیز مکان توپ از سرور گرفته شده است . این تابع به دو قسمت اصلی تقسیم شده است : قبل از شروع بازی و بعد از شروع بازی .

قبل از شروع بازی بازیکنان طبق آرایشی که در فایل Formation.conf به آنها داده اید و توسط تابع teleportToPos() در زمین قرار میگیرند .

- تابع teleportToPos تنها یک بار و در ابتدای بازی مورد استفاده قرار می گیرد . در غیر اینصورت خطای داور انسانی گرفته می شود .

بعد از شروع بازی ، بازیکنان باید بر اساس دستورات داده شده بازی کنند . در این بیس در ابتدا تنها کاری که بازیکنان انجام می دهند ، شوت کردن توپ به سمت دروازه حریف می باشد . توابعی که این اکت ها را انجام می دهد درون فایلی به نام basicPlayer.cpp قرار دارد

**توابع در بیس UVA_Trilearn :**

برای کد نویسی در محیط بیس، نیازمند استفاده از توابع موجود برای پیاده سازی الگوریتم هایتان هستید. برای مثال برای بدست آوردن حالت بازی، موقعیت بازیکن خودی، موقعیت توپ و ... میتوانید از توابع آماده در بیس استفاده نمایید.

اکثر توابع مورد نیاز شما، در فایل های WorldModel.h,BasicPlayer.h,SoccerTypes.h,Geometry.h تعریف شده اند. برای مثال برای استفاده از توابع مربوط به هندسه، مثل دایره، مخروط و ... باید در کتابخانه Geometry.h به دنبال تابع مورد نظر بگردید. طریقه یافتن توابع هم بدینگونه است که شما باید ابتدا کلمات مرتبط با موضوع مورد نظرتان را بصورت انگلیسی پیدا کرده و سپس در این فایل جستجو کنید. برای مثال اگر میخواهید تابع مربوط به کشیدن دایره را پیدا کنید بهترین راه این است که واژه circle به معنی دایره را در کتابخانه Geometry.h جستجو کنید. با این کار کلیه توابعی که با دایره در ارتباط هستند را میابید و تابع مورد نظرتان را انتخاب مینمایید. اگرچه دامنه توابع در این بیس بالاست، ولی همیشه جوابگوی شما نیست. مثلا وقتی میخواهید الگوریتمی نظیر

پاس بنویسید، نیازمند ایجاد توابع برای سهولت در پیاده سازی آن هستید. شما میتوانید توابع مورد نیازتان را به دو صورت  تعریف نمایید :

۱ – توابع را در کتابخانه های موجود در بیس تعریف نمایید.

مثال : هم اکنون میخواهیم تابعی برای الگوریتم پاس بنویسیم. ابتدا در BasicPlayer.h نام تابع را تعریف میکنیم و ورودی هایش را مینویسیم(مثلا funcName(int firstParam, int secondParam) ). سپس در فایل BasicPlayer.cpp بدین صورت تابع را مینویسیم :

```
BasicPlayer::funcName(int firstParam, int secondParam)

{

// Your Algorithm

}
```

حال برای استفاده این تابع و اجرای آن در روند کار اصلی، باید فایل BasicPlayer.h را در فایل PlayerTeams.cpp، include کرده ( #include “BasicPlayer.h” ) و در مکان های مورد نیاز، فقط نام تابع را نوشته و از آن استفاده نمایید.

۲ – شما میتوانید با ایجاد فایل ها در بیس و اضافه کردن آدرس آنها در makefile.am، از کتابخانه های خودتان استفاده نمایید.

برای ارسال دستورات اجرایی به سرور باید از کامند زیر استفاده کنید :

```
ACT->putCommandInQueue(/* your command*/  );
```

- در مقاله بعدی، راجع به ساختار PlayerTeams بصورت مفصل توضیح داده خواهد شد.

بعد از کدنویسی، برای ایجاد تغییرات و یا احیانا ارور های موجود در کد، باید در ترمینال از دستور make all استفاده نمایید. با این دستور، کلیه فایل ها کامپایل میشود و در نهایت اگر اروری در کد ها نباشد، تغییرات ذخیره سازی میگردد. اگر هم ارور موجود باشد، در ترمینال، مکان دقیق آن نمایش داده میشود و تغییری در روند بازی تیم ایجاد نمیشود و تیم به همان صورتی که قبل از آخرین کامپایل مجدد صورت گرفته بود، بازی میکند.

پایان جلسه چهارم