بسم الله الرحمن الرحيم

UVA Trilearn Base راهنمای

University of Amsterdam, The netherlands
base code version of the RoboCup-2003 World Champion

Created by:	Jelle Kok
Research Coordinator:	Nikos Vlassis
Team Coordinator:	Frans Groen
Editor:	Mohammad Ali Taghvazadeh
Translation :	Seyyed Muhammed Sadegh Salehi

اطلاعات کلی

سورس منتشر شده حاوی پیاده سازی سطح پایین و متوسط (متد همزمان سازی محیط ایجنت، WorldModel و مهارت های بازیکن) است؛ اما شامل روش تصمیم گیری سطح بالایی که خود ما استفاده می کنیم نمی باشد. در عوض، ما نمونه ای سطح بالا از استراتژی انتخاب عمل که همانند نمونه از تیم پرتغال است، اضافه کردیم. سریعترین بازیکن به توپ، آن را گرفته و بدون در نظر گرفتن موقعیت خود در میدان به سمت دروازه حریف شوت می کند. بقیه بازیکنان به موقعیت استراتژیک خود که توسط موقعیت مبدا آن ها در چیدمان و موقعیت توپ مشخص می شود؛ حرکت می کنند.

علاوه بر این، ما عمدا بعضی از قسمت های کد (سطح – پایین) خود را حذف کردیم؛ مانند : یادگیری مهارت دریبل و مدل سازی مهارت ره گیری توپ توسط بازیکنان حریف. قصد ما این بود که اطمینان حاصل کنیم سورس کد بیس ما نقطه ای مناسب (و نه خیلی عالی)برای شروع باشد؛ از آنجا که برای تیم هایی که سال های اخیر در حال کار کردن بر روی پایه خود هستند ناعادلانه خواهد بود و تیم های جدیدکه از کد مابه عنوان پایه استفاده می نمایند بلافاصله از آن ها پیشی خواهند گرفت.

کاربرد

Autconf و Automake به منظور ساخت این پکیج استفاده شدند. کد های زیر به سادگی سورس را کامپایل خواهند کرد:

```
./configure
./make
```

سورس تحت لینوکس توسعه یافته، که پلتفرم پیشنهادی است. هر چند Alexey Vasilyev یک پورت ویندوز برای کامپایلر رایگان Borland C[1] فراهم کرده است.

دایرکتوری windows/. را برای میک فایل غیر ضروری و فایل های پیکربندی بورلند نگاه کنید.

بعد از آنکه باینری ها کامپایل شدند، می توان با اسکریپت شروع در 'start.sh' از آن استفاده کرد. (محتویات این فایل را برای جزئیات نگاه کنید.)

برای گسترش استراتژی سطح بالا از این تیم، به متد 'deMeer5()' نگاه در فایل PlayerTeams.cpp که شامل استراتژی سطح بالای این تیم است نگاه کنید. قسمت های خیلی مهم دیگر برای بهبود بخشی عبارتند از:

- رفتار دروازه بان
- متد ره گیری که حریف را در نظر می گیرد.
- مهارت دریبل

مستندات

سورس کد به طور گسترده ای با استفاده از Doxygen[2] مستند سازی شده است. مستندات html ساخته شده از سایت ما قابل دانلود هستند و یا توسط استفاده از دستور زیر در پوشه اصلی سورس کد ساخته می شوند :

```
make doc
```

فایل های html در دایرکتوری doc/html/. قرار خواهند گرفت. در اینجا فرض بر این بوده است که برنامه 'dot' به منظور ساختن

1 Borland C++
2 Www.doxygen.org

دیاگرام کالیبریشن از قبل نصب شده باشد.
اگر می خواهید به این صورت نباشد، متغیر 'HAVE_DOT' در فایل doc/doxygen.cfg را به 'NO' تغییر دهید.

تغییرات

کد پایه (Basic Code) با احترام گذاشتن نسبت به کد پایه قبلی منتشر شده در سال ۲۰۰۲ (و نه تنها به همراه فایل هایی که اکنون به پسوند cpp ختم می شوند) تغییر یافته است؛ بجز برای بهبود بخشی در متد های سطح پایین، برای مثال : ره گیـــری (Intercept) و شوت (Kick) و ساختار WorldModel . سورس کد همچنین از قابلیت پروتکل های ۹ بـه بعـد پشـتیبانی مـی کنـد. بـرای مثـال : tackle, attentionto , Synchronization mode

ما سعی کردیم تا سورس کد را سازگار با نسخه های قبلی نگاه داریم، اما ما هیچ تعهدی بر دوش نمیگیریم.لطفا هنگامی که به هر مشـکل عمده ای بر خورد کردید، به ما گزارش دهید.

تقدیرنامه ها:

این تیم ابتدا در سال ۲۰۰۱ برای پروژه کارشناسی ارشد Jelle Kok و Remco de Boer در دانشگاه امستردام بوجود آمد. بعد از آن Jelle Kok این کار را تحت نظارت Nikos Vlassis ادامه داد.
اگر چه ما به هیچ وجه از تیم های دیگر کد کپی نکرده ایم، به بعضی از متد های آنها نگاه کرده و دانش آنها را برای پیاده سازی خود بـــه کار گرفته ایم. برای همین، ما می خواهیم از تیم های زیر تشکر کنیم :

- FC Prtugal 2000 : برای چیدمان و نمونه تیم آن ها
- CMUnited-99 : برای ره گیری (Interception) و متد های تجزیه پیام آن ها
- Cyberoos 2000 : برای توضیحات متد همزمان سازی آن ها
- Essex Wizards : برای توضیحات معماری multi-thread آن ها

توضیحات بیشتر

توضیحات بیشتر در وب سایت رسمی UvA Trilearn قابل دسترسی هستند :

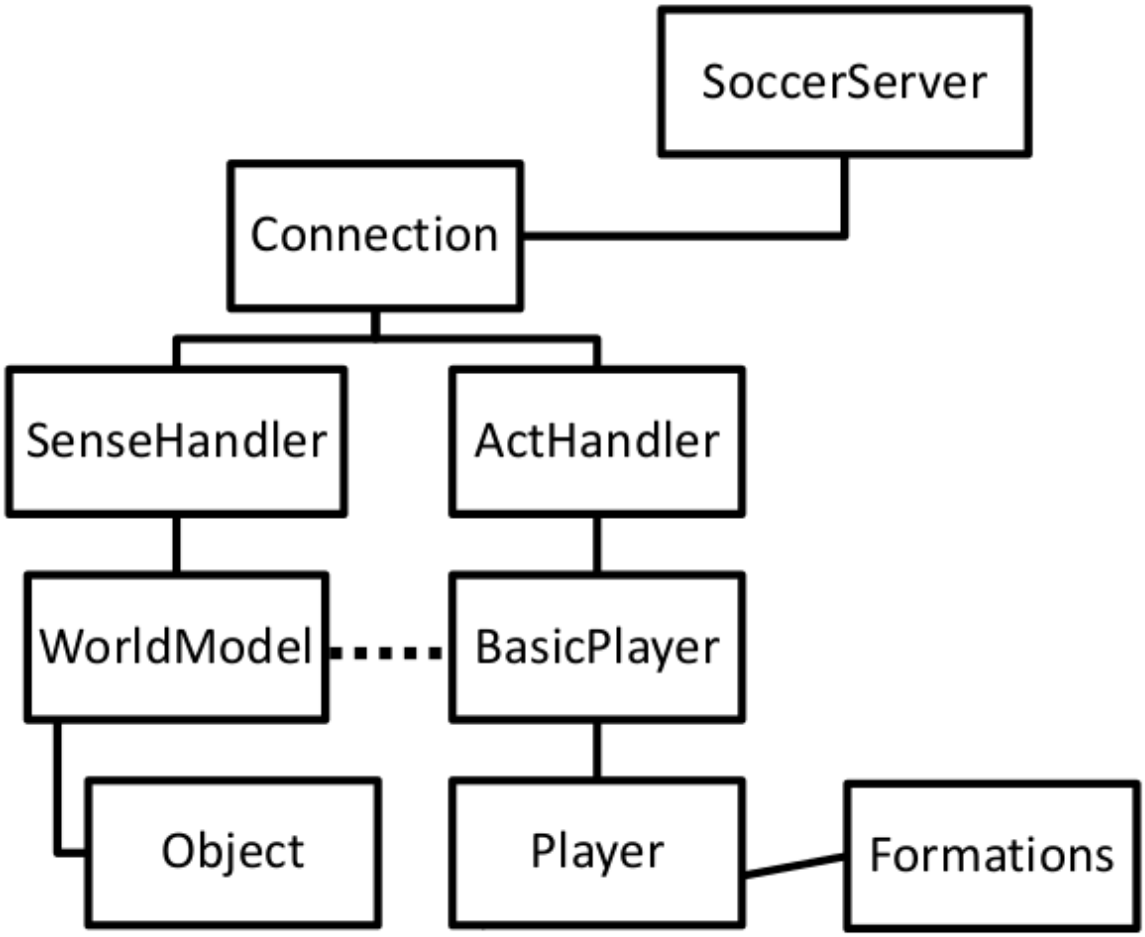
http://www.science.uva.nl/~jellekok/robocup/index.html

و یا تماس با :

Jelle Kok (jellekok@science.uva.nl)
SMSS(translator) (salehi1994@gmail.com)

مرور کلی کلاس ها

مرور کلی از کلاس های مختلف در زیر نشان داده شده اند. توجه داشته باشید کــه SenseHandler, ActHandler و کلاس هــای دیگر به سه شکل مختلف دیگر به طور مستقل کار می کنند.



کلاس های مفید دیگری که توسط کلاس هایی که در دیاگرام بالا مشاهده کردید استفاده می شوند ولی در آن نیستند، در ادامه امده اند :

- PlayerSettings
- Logger, Timing (in Logger.cpp)
- ServerSettings
- SoccerTypes, SoccerCommand, Time(in SoccerTypes.cpp)
- Geometry, Line, Circle, Rectangle, VecPosition (all in Geometry.cpp)
- Parse

توضیحات کلاس ها

یک توضیح برای هر کلاسی در زیر داده شده است :

Connection

این کلاس یک اتصال (Connection) به همراه سوکت (socket) ایجاد می کند که شامل متد هایی برای ارسال و دریافت پیام ها از این سوکت است.

SenseHandler

این کلاس پردازش پیام هایی که ایجنت از سرور دریافت می کند را بر عهده دارد. پیام ها را تجزیه کرده و اطلاعات دریافتی را به WorldModel انتقال می دهد. همچنین یک سیگنال را برای تشخیص وقتی که عمل ای (action) باید به سرور ارسال شود تنظیم می کند؛ این سیگنال توسط ActHandler برعهده گرفته می شود.

ActHandler

این کلاس با دو فعال کننده خروجی تعامل می کند؛ اعمال (Actions) را در دو صف مختلف قرار می دهد:

- m_queueOneCycleCommand:

  شامل دستوراتی که تنها یکبار در یک سایکل می توانند اجرا شوند؛ مانند Kick, Dash و غیره

- m_queuMultipleCommands:

شامل دستوراتی که می توانند همزمان با دستورات داخل m_queueOneCycleCommand هستند اجرا شوند؛ مانند : turn_neck, say و غیره؛ همه دستورات این لیست به سرور ارسال می شوند.

وقتی ActHandler سیگنالی دریافت می کند، آن را از SoccerCommand به یک رشته (string) تبدیل می کند و آن را به سرور ارسال می کند.

WorldModel

این کلاس شامل نماینده از آنچه اکنون توسط ایجنت مشاهده می شود است. این نماینده شامل اطلاعات درباره تمامی اشیاء (Objects) که در میدان هستند مانند مختصات و سرعت ها از همه بازیکنان و توپ است. اطلاعات در باره حالت بازی کنونی (current play mode) نیز ذخیره می شوند، به همان صورت که زمان و امتیاز ذخیره می شوند. علاوه بر این، WorldModel شامل انواع مختلفی از متدهای تعامل با وضعیت اطلاعات جهان از راه های مختلف است :

- متد های تشخیص (Retrieval methods) :

  به منظور تشخیص مستقیم اطلاعات در رابطه با یک شئ (object) در World Model ؛ این متد ها در WorldModel.cpp تعیین شده اند. این فایل همچنین شامل متد هایی برای تکرار کردن (iterate) یک نوع از اشیاء است. این متد ها این امکان را فراهم می آورند که اطلاعات اشیاء مختلفی در یک نوع را با هم مقایسه کرد. برای مثال در نوع : OBJECT_SET_OPPONENTS

- متدهای به روز رسانی(Update methods) :

  برای به روز رسانی WorldModel بر مبنای اطلاعات حسی دریافت شده از SenseHandler پایه گذاری شده است. این متد ها در فایل WorldModelUpdate.cpp تعیین شده اند.

- متد های پیش بینی (Prediction methods) :

  برای ویژگی پیش بینی، وضعیت جهان اطراف بر مبنای ادراک گذشته و اثر اعمال اجرا شده توسط ایجنت است؛ این متد ها در فایل WorldModelPredict.cpp تعیین شده اند.

- متد های سطح بالا (High-level methods) :

برای استخراج سطح بالا نتیجه از اطلاعات پایه درباره وضعیت جهان اطراف است. برای مثال : تعیین سریعترین هم گروه به توپ؛ این متدهای در فایل WorldModelHighLevel.cpp تعیین شده اند.

Object

این کلاس شامل اطلاعات درباره همه اشیاء در شبیه سازی است.پیاده سازی آن توسط بیش از شش کلاس مجزا گسترش یافته که با یکدیگر سسله مراتب نوع شئ را تشکیل می دهند. این کلاس ها در زیرآمده اند:

- Object

ابر کلاس انتزاعی که شامل تخمین ها (و مقادیر اطمینان) برای موقعیت جهانی (Global Position) از همه اشیاء است و متد به روز رسانی آن ها را تعیین می کند.

- FixedObject

زیر شاخه کلاس Object که شامل اطلاعات درباره اشیاء ثابت در میدان (پرچم ها (flags), خطوط (lines) و دروازه ها (goals) است؛ این کلاس خواص فرعی اشیاء را که از ابر کلاس Object به ارث برده شده اند را نمی افزاید.

- DynamicObject

زیر شاخه کلاس Object که شامل اطلاعات در باره اجسام در حال حرکت است؛ این کلاس اطلاعات سرعت را به اطلاعات عمومی آن ها که توسط کلاس Object ارائه می شود را می افزاید.

- BallObject

زیر شاخه کلاس DynamicObject که شامل اطلاعات در باره توپ است. این کلاس خواص فرعی که از DynamicObject به ارث برده شده اند را نمی افزاید.

- PlayerObject

زیر شاخه کلاس DynamicObject که شامل اطلاعات در باره یک بازیکن خاص در میدان (چه همگروه و یا حریف) است.این کلاس خواص بیانگر زاویه جهانی گردن و بدن بازیکن را به اطلاعاتی که توسط DynamicObject ارائه شده اند می افزاید و این کلاس شامل خاصیت یک مقدار منطقی (Boolean) که مشخص می کند بازیکن دروازه بان است یا نیست؛ و ایجنت اجرا کننده عضو این کلاس نیست.

- AgentObject

زیر شاخه کلاس PlayerObject که شامل اطلاعات درباره خود ایجنت است. همچنین خواص بیانگر استقامت (stamina)، زاویه دید (view angle) و کیفیت دید (view quality) ایجنت از اطلاعات ارائه شده توسط کلاس PlayerObject است.

BasicPlayer

این کلاس مشخص کننده مهارت های گوناگون که ایجنت قابلیت اجرای آن ها را دارد است. راهی که این مهارت ها در آن اجرا می شوند منوط به وضعیت کنونی از جهان اطراف (current state of world model) است.

PlayerSettings

این کلاس شـــامل پارامترهـایی اسـت کـه در کلاس BasicPlayer اسـتفاده شـده انـد. یـک مثـــال از چنیـن پـارامتر هـایی 'dPassEndSpeed' است که بیانگر سرعت پایانی مطلوب توپ وقتی که به همگروه پاس داده می شود است. توسط تغییر این مقـدار از پارامتر ها که در این کلاس است؛ این امکان وجود دارد که یک رفتار BasicPlayer را تغییر(وفق) داد.

Player :

این کلاس یک زیر شاخه از BasicPlayer است که شامل متد هایی برای استدلال درباره بهترین عمل ممکـن در وضـعیت داده شـده است. انتخاب کردن عمل بر مبنای اخرین اطلاعات در رابطه با وضعیت کنونی جهان که از WorldModel مشـاهده مـی شـود و نقـش بازیکن در چیدمان کنونی تیم است. برای ساختن تصمیم نهایی در مورد اینکه چه نوع خاصی از عمل باید اجرا شود، ایجنت از پارامتر مقدار ها که درکلاس PlayerSettings مشخص شده اند استفاده می کند.

Formations

این کلاس شامل اطلاعات درباره امکان چیدمان (Formation) تیم به صورت همزمان با متد تعیین مـوقعیت اسـتراتژیک در میـدان است. چیدمان ها از یک فایل پیکر بندی خوانده می شوند (formations.conf) و بر همان مبنای نمونه تیم پرتغال هستند. پیاده سازی توسط بیش از سه کلاس جداگانه گسترش یافته است:

- PlayerTypeInfo :

  شامل اطلاعات درباره نوع بازیکن در چیدمان

- FormationTypeInfo :

  شامل اطلاعات درباره یک چیدمان مشخص

- Formations :

شامل اطلاعات درباره همه امکان چیدمان های تیم و ذخیره حالت استفاده چیدمان کنونی اســت؛ و ازکلاس WorldModel قابل دسترسی است.

GenericValues

این کلاس یک ابر کلاس برای همه کلاس هایی که شامل تنظیمات ازکلاس های PlayerSettings و ServerSettings می شود است. استفاده از این کلاس است که متغیر ها را به نام آن ها تغییر می دهد. وقتی این نام ها به همراه مقادیر مربوط به آن ها از فایلی خوانده (و یا در آن نوشته) می شوند، مقادیر مربوطه به آسانی می توانند تنظیم شوند و نیازی نیست که برای تغییر یک متغیر در سراسر برنامه بـه دنبال آن ها باشیم.

Logger

این کلاس توسط همه ی کلاس های دیگر برای ثبت وقـایع (Logging) انـوع مختلـف اطلاعـات بـرای مقاصـد اشـکال زدایـی (debugging)، استفاده می شود. این کلاس به برنامه نویس اجازه می دهد تا سطح انـتزاعی ثبـت وقـایع (log-level) را کـه شـامل اطلاعات مطلوب برای اشکال زدایی و جریان خروجی برای نوشتن (معمولا در یک فایل) می شود را مشخص کند. تمامی اطلاعاتی که بـه Logger ارسال می شوند یک شماره دارند که با سطح مشخصی مقایسه می شوند که تعیین شود که ایا اطلاعات باید چاپ شوند و یـا بایـد نادیده گرفته شوند. همچنین این امکان وجود دارد که اطلاعات به همراه تمبر زمان ثبت گردند. این زمان تمبر مربوط به زمانی است کـه از آخرین زمان شروع مجدد زمان سنج سپری شده است. این زمان سنج نشان داده شده است توسط شئ از کلاس Timing کـه همچنیـن در فایل 'Logger.cpp' تعیین شده است.

Timming

این کلاس شامل زمان سنج و متد هایی برای شروع مجدد و تعیین مقدار زمان ساعت سپری شده از وقتی که زمان سنج شروع به کار نموده است که عمدتا برای پیام های ورودی از سرور و برای مقاصد اشکال زدایی به کار می رود.

Parse

این کلاس شامل تعداد زیادی متدهای ثابت برای تجزیه رشته پیام ها است. این متد ها قابلیت پریدن از روی کاراکتر ها تا نقطه ای مشخص و تبدیل قسمت هایی از یک رشته به مقدار صحیح (int) و یا double را دارا می باشد. آن ها عمدتا توسط SenseHandler که وظیفه پردازش پیام های از سرور فوتبال را دارد مورد استفاده قرار می گیرند.

ServerSettings

این کلاس شامل همه پارامترهای سرور است که برای نسخه کنونی از سرور فوتبال استفاده شده است. مثال: نهایت سرعت بازیکن (player_speed_max) و افزایش استقامت در هر ثانیه (stamina_inc_max).

وقتی که ایجنت مقدار دهی اولیه می شود، سرور پیام مقدار ها را برای او ارسال می کند که شامل مقدارهای این پارامتر های هستند. این پیام پس از تجزیه شده توسط متدهایی از کلاس Parse و نتیجه گیری مقدار ها، در ServerSettings ذخیره می شوند.

SoccerTypes

این کلاس شامل انواع داده های شمارشی (enum) برای انواع مختلف فوتبال است که در شبیه سازی استفاده می شوند. این کلاس یک تجرید برای استفاده در مفاهیم مرتبط فوتبال ایجاد می کند؛ مانند : حالت های بازی (playmodes) ، پیام های داور (referee messages) و غیره. این enum باعث خوانایی برنامه در حالتی تمیز و استوار در سرتاسر کد می شود. علاوه بر این، این کلاس شامل متدهای تبدیل قسمت های یک رشته پیام دریافت شده از سرور به soccerTypes را دارد. برای مثال : '(g l)' را به '(GOAL_LEFT)' می تواند تبدیل کند.

SoccerCommands

این کلاس نگه دارنده تمام اطلاعات ضروری برای ساخت یک دستور فوتبال (soccer command) است که قابلیت ارسال به سرور را دارد. این کلاس شامل متغیر هایی است که بیانگر آرگومان های ممکن مانند زاویه، قدرت و غیره از دستور های مختلف فوتبال هستند و نوع دستور کنونی را ذخیره می کند. فقط آن دسته از متغیر هایی که مربوط به نوع کنونی هستند یک ارزش قانونی خواهند گرفت. بعلاوه، این کلاس شامل متد برای تبدیل دستور (command) به یک رشته (string) پیام است که توسط سرور فوتبال پذیرفته خواهد شد. تعریف این کلاس را در فایل 'SoccerTypes.cpp' قابل دسترسی است.

Time

این کلاس زمان سرور در شکل زوج مرتب (t,s) نگه داری می کند که t بیانگر سایکل کنونی سرور (current server cycle) و s بیانگر تعداد سایکل هایی است که از آن زمان ساعت متوقف شده است. در اینجا مقدار t برابر مقدار تمبر زمان است که شامل آخرین زمان دریافت شده از سرور می باشد، در حالیکه مقدار s همیشه برابر با صفر خواهد بود در زمانی که بازی در حال اجرا باشد. s فقط در حالت هایی که توپ مرده باشد (برای مثال ضربه های آزاد) که متغیر خواهد بود از زمانی که این حالات باعث توقف زمان سرور به صورتی که بتواند از سر گرفته شود باشد.

Geometry

این کلاس شامل متد های ثابت بسیاری برای اجرا کردن محاسبات هندسی است و عمدتا توسط BasicPlayer برای کار کردن با جزئیات عمل (action) مورد استفاده قرار می گیرد. متدها برای تعامل با (حتی بی نهایت) دنباله های هندسی و کار با فرمول خطی تعیین شده اند. در نظر بگیرید که فایل Geometry.cpp همچنین شامل بسیاری تابع جهت یابی(goniometric) است که قادر به

مشخص کردن یک زاویه مشخص به درجه و یا رادیان است.

VecPosition

این کلاس شامل نماینده ای از موقعیت (مختصات) x,y است که متد های بسیاری را تعیین می کند که بر روی این مختصات در راه های مختلفی عمل نمایند. متد ها برای مقایسه نسبی مختصات.(برای مثال:'isBetween' ، 'isBehind' و غیره) و برای تبدیل کردن مختصات به مختصات جهانی و برعکس کردن آن معین هستند. این کلاس همچنین به شما اجازه می دهد تا مختصات را به صورت قطبی (r,phi) مشخص کنید و شامل یک متد دیگر برای تبدیل مختصات قطبی (r, phi) به مختصات دکارتی (x,y) است. علاوه بر این، عملگر های حساب استاندارد برای مختصات ها مجددا بارگذاری شده اند. تعریف این کلاس در فایل 'Geometry.cpp' قابل دسترسی است.

Line

این کلاس شامل نماینده ای یک خط : $ax + by + c = 0$ است که به شما اجازه می دهد یک خط را در راه های مختلفی اعلان کنید. توسط ارائه کردن سه مقدار (a,b,c)، توسط دادن دو نقطه از یک خط، و یا توسط یک نقطه و یک زاویه مرتبط با آن. علاوه بر این، این کلاس شامل متد هایی برای تعیین نقطه برخورد دو خط و برای تعیین عمود یک خط که توسط خط موجود و نقطه ای که به آن داده می شود. تعریف این کلاس در فایل 'Geometry.cpp' قابل دسترسی است.

Cirlce

این کلاس شامل نماینده از دایره و متدهایی که با دایره تعامل دارند است. یک دایره توسط یک شئ VecPosition که بیانگر مرکز دایره است و یک مقدار که بیانگر شعاع آن است مشخص می شود. متد هایی برای تعیین محاسبات مساحت و محیط این دایره و تقاطع دو دایره به علاوه بر مساحت آن ها نیز تعیین شده است. تعریف این کلاس در فایل 'Geometry.cpp' قابل دسترسی است.

Rectangle

این کلاس شامل نماینده از مستطیل است و شامل متد هایی برای تعامل با مستطیل می باشد. یک مستطیل توسط دو شئ از VecPosition مشخص می شود که به ترتیب بیانگر نقطه بالا-چپ و پایین-راست مستطیل هستند. مهم ترین متد این کلاس مشخص کننده این است که ایا نقطه داده شده درون مستطیل نهفته است یا خیر. تعریف این کلاس در فایل 'Geometry.cpp' قابل دسترسی است.

فایل اصلی :

در فایل اصلی 'main.cpp' تمامی کلاس های گوناگون مقدار دهی اولیه می شوند و بعد از اینکه حلقه بازیکن فراخوانی شد، پیوند می خورند. یک واحد منفرد اجرایی از این حلقه همانند زیر است :

- مسدود کردن (block) تا زمانی که اطلاعات حسی جدید دیگری دریافت شود.
- دستور دادن به WorldModel به منظور به روز رسانی اطلاعات با استفاده از آخرین پیام دریافتی از سرور
- مشخص کردن بهترین عمل ممکن در شکل یک مهارت از کلاس Player
- ارسال یک دستور عمل به ActHandler که قسمت از این مهارت است

این حلقه فراخانی می شود در هر سایکل بعد از حس-بدنی (sense-body) و یا مشاهده اینکه پیامی دریافت شده باشد. بعد از یک حس-بدنی پیام، یک عمل جدید بر مبنای پیش بینی از حالت کنونی جهان اطراف مشخص می شود؛ در حالیکه بعد از دیدن پیام، عمل جدید برگزیده بر مبنای اطلاعات بصری جدید خواهد بود. در هر صورت، عمل در صف ActHandler قرار خواهد گرفت. در این راه ActHandler همیشه شامل یک اکشن خواهد بود که اطلاعات وتا زمانی که اطلاعات بصری در یک سایکل نرسند (دریافت نشوند).

وقتی که اطلاعات بصری (visual information) می رسند (دریافت می شوند)، عمل (action) مشخص شده بعد از رسیدن پیام–حسی می تواند بر مبنای اطلاعات جدید بهینه سازی شود، اما این کار ضروری نیست.

پاییز ۱۳۹۲

پایان