دانشگاه تهران

دانشکده‌ی مهندسی برق و

کامپیوتر

گزارش نهایی پروژه سیستم‌های سایبر-فیزیکی

اپلیکیشن بازی Air Hockey

اعضای گروه:

هومان چمنی, علیرضا سالمی , رضا قنبری و مهدی صالحی

شماره دانشجویی:

۸۱۰۱۹۶۴۴۳ , ۸۱۰۱۹۶۴۸۰ , ۸۱۰۱۹۶۵۲۸ و ۸۱۰۱۹۶۵۰۰

اساتید:

دکتر مهدی کارگهی ، دکتر مهدی مدرسی

ترم بهار ۱۴۰۰

فهرست مطالب

۱- مقدمه

ساختن بازی‌های جذاب و سرگرم‌کننده برای تلفن همراه که با هزینه پایین و توان مصرفی قابل قبول قابل استفاده باشند می‌تواند کمک به سزایی به پر کردن اوقات فراغت افراد کند. یکی از این بازی‌های بازی دو نفره به نام Air Hockey است که به صورت بی‌درنگ[1] روی دو دستگاه تلفن همراه با سیستم‌عامل اندروید اجرا می‌شود. به منظور استفاده بهینه تر از انرژی و امکان برقراری ارتباط پایدارتر از ماژول بلوتوث برای ارتباط دو دستگاه استفاده می‌شود. در این بازی هر یک از طرفین کنترل دسته خود را بر عهده دارد و به دیسک در یک زمین بدون اصطکاک ضربه می‌زند و قصد دارد که توپ را وارد دروازه حریف نماید. هرکس که زودتر موفق به زدن ۷ گل شود برنده بازی است .همچنین لازم به ذکر است که در این پیاده‌سازی از بازی Air Hockey از موتورهای بازی‌سازی معمول استفاده نشده و طراحی و پیاده‌سازی از پایه انجام می‌شود.

- **محدوده پروژه**

محدوده این پروژه شامل انجام پیاده‌سازی در محیط کامپیوتر و اجرای واقعی خروجی روی دو دستگاه با سیستم عامل اندروید می‌باشد. عناوین اصلی و مهمی که در محدوده‌ی این پروژه قابل تصور هستند شامل موارد زیر می‌شوند:

* ساختار و نحوه کار با ماژول بلوتوث در سیستم عامل اندروید
* ایجاد نوعی ارتباط بی‌درنگ از دید کاربران سیستم
* آشنایی با مفاهیم ساخت برنامه برای سیستم عامل اندروید
* آشنایی با فیزیک حاکم بر بازی‌هایی که مبتنی بر برخورد اشیا[2] می‌باشند
* نحوه تولید برنامه‌ای که قابلیت پشتیبانی از چند نسل سیستم عامل اندروید را داشته باشد

- **اهداف پروژه:**

صنعت بازی و سرگرمی روی دستگاه‌های تلفن همراه در دهه‌های اخیر بسیار مورد توجه قرار گرفته است. همه ساله نیز با پیشرفت تلفن‌های همراه و افزایش قابلیت‌های سخت‌افزاری و نرم‌افزاری آنان این توجه گسترده‌تر

[1] Real time
[2] Objects

می‌شود. هدف اصلی این پروژه ارائه یک بازی به صورتی است که بازیکنان می‌توانند با فاصله مشخصی از یک دیگر بر روی دو دستگاه مجزا که دارای سیستم‌عامل اندروید هستند بازی را انجام دهند. همچنین به منظور این که مصرف انرژی در این بازی کم باشد، برای انتقال اطلاعات و همگام‌سازی دو دستگاه و وضعیت بازی بین آن‌ها از ماژول بلوتوث برای برقراری ارتباط بین دو دستگاه استفاده می‌شود. از دلایل انتخاب این پروژه می‌توان به موارد زیر اشاره کرد:

* جامعیت موضوع
* استفاده از ماژول بلوتوث که شامل مراحل یافتن[3], متصل شدن[4] و انتقال اطلاعات می‌باشد
* در نظر گرفتن مباحث بی‌درنگ به منظور بالا نگه داشتن رضایت کاربران
* کار با سیستم عامل اندروید و همچنین مرتبط بودن با صنعت بزرگ سرگرمی

۲- معرفی پلتفرم و ابزارهای استفاده شده در پروژه

همان‌طور که انتظار می‌رود, تمامی پیاده‌سازی‌های مربوط به این بازی در محیط Android studio انجام شده است. همچنین لازم به ذکر است که از نسخه ۴.۲.۱ این برنامه در جریان پروژه استفاده شد. یکی از مهم‌ترین بسته‌هایی[5] که در این پروژه استفاده شد بسته android.bluetooth می‌باشد. این بسته تمامی API های کار کردن با ماژول بلوتوث را در خود شامل شده و قابلیت‌های پایه‌ای آن شامل موارد زیر می‌شوند:

* یافتن دستگاه‌های بلوتوثی نزدیک که یا تا کنون به دستگاه ما متصل شده یا جدید هستند
* متصل شدن به یک دستگاه بلوتوثی
* رد و بدل کردن اطلاعات با دستگاه مذکور

به منظور بالا آوردن فضای اصلی بازی و مشاهده طراحی‌های انجام شده نیز از سیستم درونی Android studio که اقدام به ساختن پروژه از روی فایل‌های آن می‌کند استفاده شد. در آخر نیز برای بررسی کارایی پروژه در محیط واقعی از دو دستگاه موبایل یا تبلت دارای سیستم عامل اندروید (نسخه بالاتر یا مساوی ۶) بهره‌برداری شد.

[3] Discovery
[4] Pairing
[5] Packages

۳- بیان چالش

به‌دلیل اینکه این پروژه یک پیاده‌سازی از یک بازی اندروید می‌باشد, لذا تعاملی با دنیای فیزیکی بیرون وجود نخواهد داشت. لذا طبیعی است که تمامی چالش‌های احتمالی این پروژه در حین پیاده‌سازی و به طور دقیق‌تر شبیه‌سازی بخش‌های مختلف مانند محیط, منطق اصلی و ارتباط بین دو دستگاه رخ دهند. در این قسمت نیز به برخی از مهم‌ترین چالش‌هایی که در این پروژه به آنان برخورد کرده‌ایم به طور خلاصه اشاره می‌کنیم:

- چالش شبیه‌سازی ضربه به توپ : مدل کردن ضربه به توپ شامل انجام محاسبات فیزیکی نسبتا پیچیده‌ای می‌شد که تا جای ممکن فرایند مشاهده شده توسط کاربران که شامل مراحلی مانند برخورد و عکس‌العمل نسبت به آن می‌شود به واقعیت نزدیک باشد.
- چالش بازیکن اصلی بودن از دید هر طرف: همان‌طور که در نسخه واقعی این بازی دیده شده‌است, هر بازیکن از سمت خود به نظر بازیکن اصلی می‌آید. به طور مثال سمت چپ هر بازیکن از دید وی سمت چپ می‌باشد اما از دید بازیکن مقابل سمت راست است. لذا برای نزدیک‌تر بودن به واقعیت و حفظ کیفیت بازی لازم بود تا هر داده‌ای که از طرف مقابل دریافت می‌شود قبل از نمایش داده شدن در صفحه بازیکن کنونی ابتدا طی انجام محاسباتی به جهت درست تبدیل شده و سپس نمایش داده شود. این الزام شامل تمامی موارد مانند دسته‌ها, توپ و مهم‌ترین عامل نیز جهت حرکت اجزا در صفحه می‌شود.
- چالش ارسال و دریافت داده: در ابتدا طبق مشاهدات انجام شده برنامه‌ریزی شده بود که نحوه ارسال داده به این صورت باشد که ابتدا داده را به صورت شی[6] تعریف کنیم و سپس این شی را به شکل بایت ارسال کنیم. چالش انجام این کار این بود که می‌بایست ابتدا یک موجودیتی به نام پیام تعریف کرده و سپس داده را در آن قرار می‌دادیم. اما لزوما پیام‌های دارای ویژگی‌های معینی نبودند و به طور مثال یک پیام دارای سه متغیر و پیام دیگری شامل پنج متغیر می‌شد. مشکلی که با تعریف کردن چند نوع پیام نیز به‌وجود می‌آمد این بود که نیاز بود قبل از خواندن پیام نوع آن را تشخیص بدهیم که سربار قابل توجهی به برنامه اضافه می‌کرد. به این دلیل بعد از مدتی تصمیم گرفته شد که برای ارسال و دریافت داده‌ها از پیام‌های متنی استفاده کنیم.

[6] Object

٤- نزدیکترین نمونه های مشابه

سه مورد از نزدیک‌ترین پروژه‌ها در این بخش بررسی شده‌اند, تمامی این موارد به صورت پروژه‌های متن‌باز و شخصی بوده و هیچ‌یک به صورت تجاری در پلتفرم‌های معمول مانند Google play منتشر نشده‌اند:

* مورد اول که نام آن نیز AirHockey می‌باشد به آدرس github در دسترس است. این پروژه با زبان جاوا و در محیط Android Studio پیاده‌سازی شده‌است. نکته‌ای که باید به آن توجه داشت این است که این پروژه از OpenGL روی محیط اندروید استفاده کرده است. طبق توضیحاتی که در مخزن گیت‌هاب مربوط به این پروژه مطرح شده, فرایند توسعه این بازی با الهام گرفتن از کتاب OpenGL ES 2 for Android انجام شده‌است.

* مورد بعدی نیز دارای نام مشابهی است. این پروژه نیز به صورت عمومی به آدرس github قابل دسترسی است. این پروژه اندکی جدیدتر بوده و دو سال بعد از مورد قبلی یعنی در سال ۲۰۱۷ پیاده‌سازی شده است. لازم به ذکر است که این بازی با استفاده از موتور بازی‌سازی Unity پیاده‌سازی شده و در نتیجه زبان‌های مورد استفاده در آن C# و Python می‌باشند. البته این بازی مانند موارد معرفی شده دیگر در آخر روی دستگاه‌های اندروید اجرا می‌شود.

* مورد آخر نیز به آدرس github قابل دسترسی می‌باشد. این پروژه در سال ۲۰۱۹ انجام شده و نکته جالب توجه درباره آن این است که محیط پیاده‌سازی آن ترکیبی از Android Studio و BBEdit مربوط به سیستم‌عامل مک[۷] می‌باشد. این پروژه نیز دارای گواهی[۸] MIT بوده که بیان می‌دارد استفاده از آن حتما باید با ذکر منبع باشد.

[7] macOS
[8] License

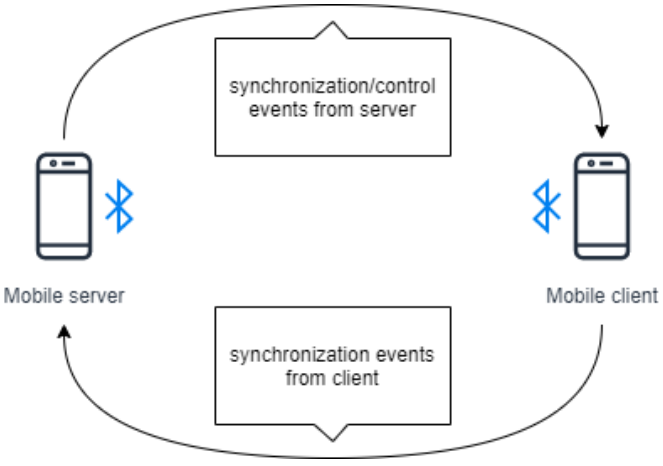
۵- مبانی فنی پروژه

○ ارائه راه‌حل پیشنهادی بصورت کلی

همان‌طور که تا کنون توضیح داده شد, این پروژه یک پیاده‌سازی از یک بازی اندروید می‌باشد. به طور دقیق‌تر یعنی یکی از مهم‌ترین مشکلاتی که این پروژه به عنوان یک راه‌حل نسبت به آن عمل می‌کند, مشکل پر کردن اوقات فراغت افراد با اپلیکیشن‌های سرگرم‌کننده است. این راه‌حل نیز به صورت یک بازی دو نفره بر بستر بلوتوث بوده که افراد دارای دستگاه اندروید بتوانند در هر زمان بدون نیاز به هرگونه اتصال به شبکه اینترنت با یکدیگر بازی کنند. به دلیل هدف ذکر شده, نحوه توسعه این بازی بدین صورت بوده است که تنها نیازمندی هر دستگاه وجود چیپ بلوتوث و داشتن سیستم‌عامل اندروید بالاتر از ۶ می‌باشد. طبق آماری که توسط وبسایت Statcounter تحت عنوان Mobile & Tablet Android Version Market Share Worldwide منتشر شده است, در ماه گذشته حدود ۸۸ درصد از کاربران اندروید در سطح جهان از نسخه ۶ یا بالاتر از این سیستم‌عامل استفاده می‌کردند. نحوه توزیع دقیق این کاربران به ازای هر نسخه در شکل ۱ قابل مشاهده است.

شکل ۱ – توزیع نسخه‌های سیستم‌عامل اندروید در ماه گذشته در سطح دنیا

قبل از اینکه به توضیحات مفصل‌تری از راه‌حل موجود بپردازیم طراحی مفهومی این پروژه را مشاهده می‌کنیم:

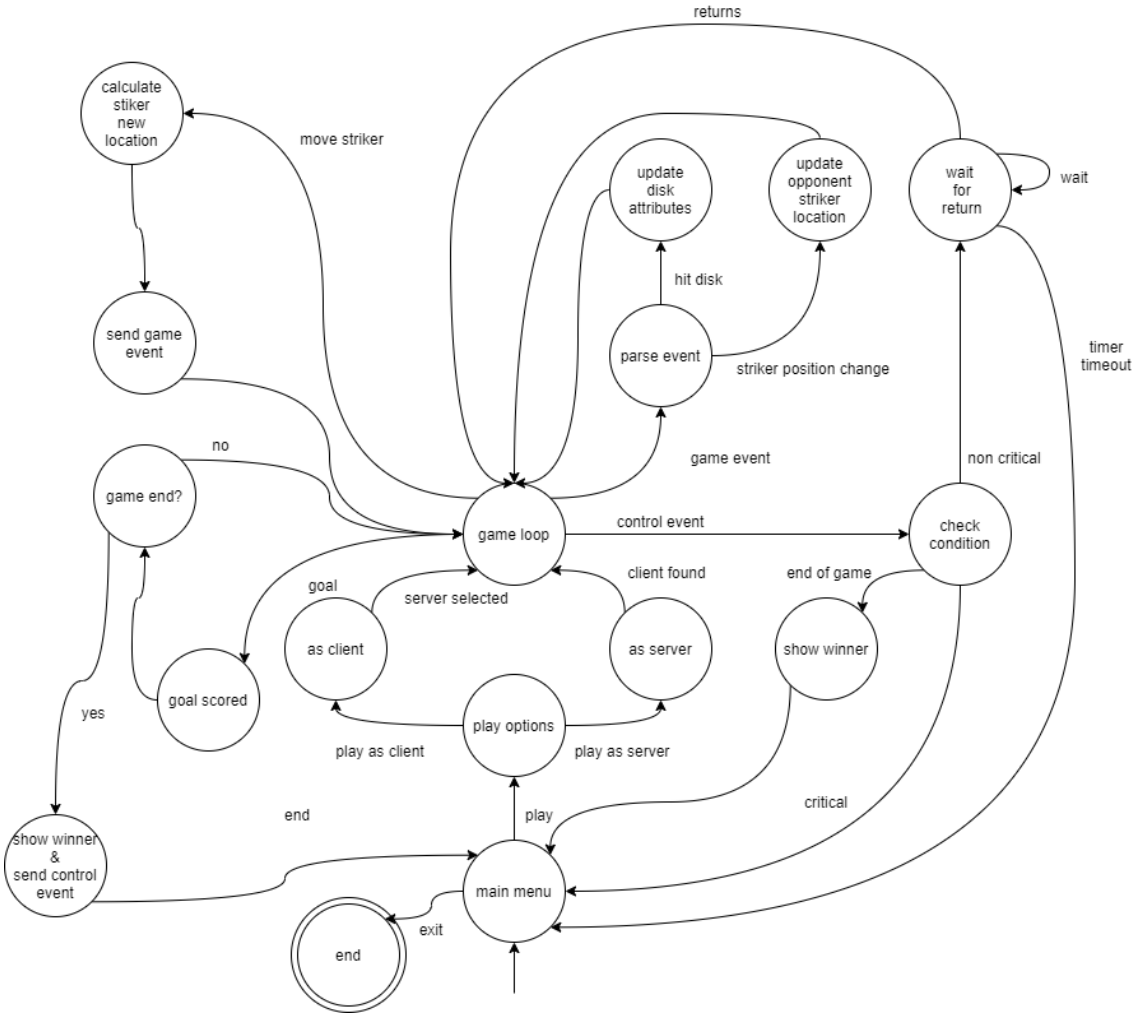


شکل ۲ – طراحی مفهومی پروژه

- **ارائه راه‌حل با جزییات**

توضیحات کامل نحوه پیاده‌سازی از جمله ساختار کد, وظیفه هر کلاس از برنامه و نحوه ارتباط اجزا با یکدیگر به طور کامل در بخش تشریح پیاده‌سازی قابل دسترسی است. در این قسمت به بیان الگورتیم اصلی برنامه می‌پردازیم:

نمودار حالت [9] برنامه:



شکل ۳ – نمودار حالت بازی Air Hockey

[9] State Diagram

در نمودار حالت فوق شروع از حالت منو اصلی است که کاربر پس از آن می‌تواند یا خارج شده یا بازی کند. اگر بخواهد بازی کند، در این صورت یا خود به عنوان سرور یک اتاق جدید ایجاد می‌کند که بازیکن دیگری به آن وارد شود یا به یکی از اتاق‌های موجود وصل می‌شود. پس از آن بازی آغاز می‌شود. به طوری کلی هر بازیکن دسته خود را جا به جا کرده و مکان جدید را به بازیکن مقابل ارسال می‌کند. در این میان اگر به دیسک ضربه‌ای زده شود نیز به طور جداگانه این اتفاق را به طرف مقابل ارسال می‌کند. از طرف دیگر تعداد رویداد نیز از طرف مقابل قابل دریافت است. این رویدادها در دو دسته کلی رویدادهای بازی و رویدادهای کنترلی قرار می‌گیرند. در رویدادهای بازی حرکت دسته طرف مقابل و ضربه به توپ به بازیکن گزارش می‌شود. در رویدادهای کنترلی گل‌های زده شده یا پایان بازی یا توقف آن به بازیکن گزارش می‌شود. برنامه در صورت دریافت هریک از این رویدادها عمل مناسب را انجام خواهد داد.

- **نحوه ی تحلیل راه‌حل و اثبات کارایی(مثلا زمان تاخیر و مصرف حافظه و ...)**

در این بازی مهم‌ترین اصل کیفیت ارتباط بین دو بازیکن است. همان‌طور که تا این بخش توضیح داده شد, هر بازیکن روی دستگاه خود به بازی با رقیب خود می‌پردازد و بحث بی‌درنگ حس شدن بازی از سمت دو طرف بسیار پراهمیت است. لذا برای تحلیل راه‌حل پیاده‌سازی شده در وهله اول به تاخیر ارسال و دریافت داده بین دو طرف خواهیم پرداخت. به دلیل ساده بودن اجزای گرافیکی بازی و کم حجم بودن محاسبات این پروژه از نظر میزان حافظه تا حد خوبی بهینه بوده و نیازی به تمرکز روی این معیار نخواهد بود.

٦- پیاده‌سازی‌های انجام شده

- **شکست کار بین اعضای تیم**

هومان چمنی:

- ❖ مطالعه و بررسی نحوه کارکرد ماژول بلوتوث و پیاده‌سازی نسخه اولیه یافتن دستگاه
- ❖ تهیه مستندات از اجرای تست کارایی برنامه و تحلیل آنان
- ❖ تهیه و نگارش گزارش نهایی پروژه

رضا قنبری:

- ❖ پیاده‌سازی نهایی ماژول بلوتوث و انتقال داده
- ❖ طراحی و پیاده‌سازی توابع لازم برای حرکت و برخورد در صفحه
- ❖ طراحی و پیاده‌سازی بخش‌هایی از رابط کاربری بازی

علیرضا سالمی:

- ❖ پیاده‌سازی توابع لازم برای حرکات درون صفحه
- ❖ طراحی منطق اصلی برنامه شامل مواردی مانند گل زدن و برد و باخت
- ❖ دیباگ کردن و تست برنامه با استفاده از دو دستگاه اندروید
- ❖ طراحی و پیاده‌سازی بخش‌هایی از رابط کاربری بازی

مهدی صالحی:

- ❖ دیباگ کردن و تست برنامه با استفاده از دو دستگاه اندروید
- ❖ مشخص سازی نحوه انجام تست برنامه
- ❖ تحلیل عملکرد برنامه در بخش تست

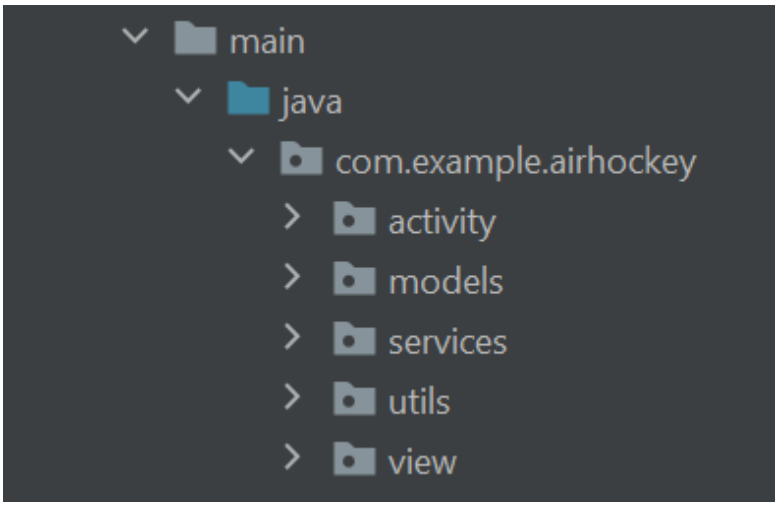
- **مشخصات محیط توسعه**

محیط توسعه این پروژه با استفاده از مشاهده اطلاعات موجود در بخش Project Structure به صورت زیر قابل توصیف است.

- نسخه افزونه [10] مربوط به Android Gradle : ۴.۲.۱
- نسخه Gradle : ۶.۷.۱
- وابستگی‌ها[11] شامل:
  - وابستگی appcompat : ۱.۳
  - وابستگی constraintlayout : ۲.۰.۱
  - وابستگی espresso-core : ۳.۳
  - وابستگی junit : ۱.۱.۲
  - وابستگی material : ۱.۲.۱

- **تشریح پیاده‌سازی(توضیحات بصورتی باشد که بعدا توسط خواننده گزارش قابل پیاده‌سازی باشد)**

کد کامل مربوط به بازی به پیوست این گزارش ضمیمه شده است. در این قسمت شرحی کلی از ساختار پروژه شامل مواردی مانند ساختار پوشه‌بندی, کلاس‌های موجود در هر بخش, وظیفه هر کلاس به‌همراه توضیح کلی متدها و ارتباط کلی کلاس‌ها را ارائه می‌دهیم. همان‌طور که در شکل زیر قابل مشاهده است, پیاده‌سازی اصلی برنامه که در بخش main قرار دارد شامل ۵ پوشه می‌شود.

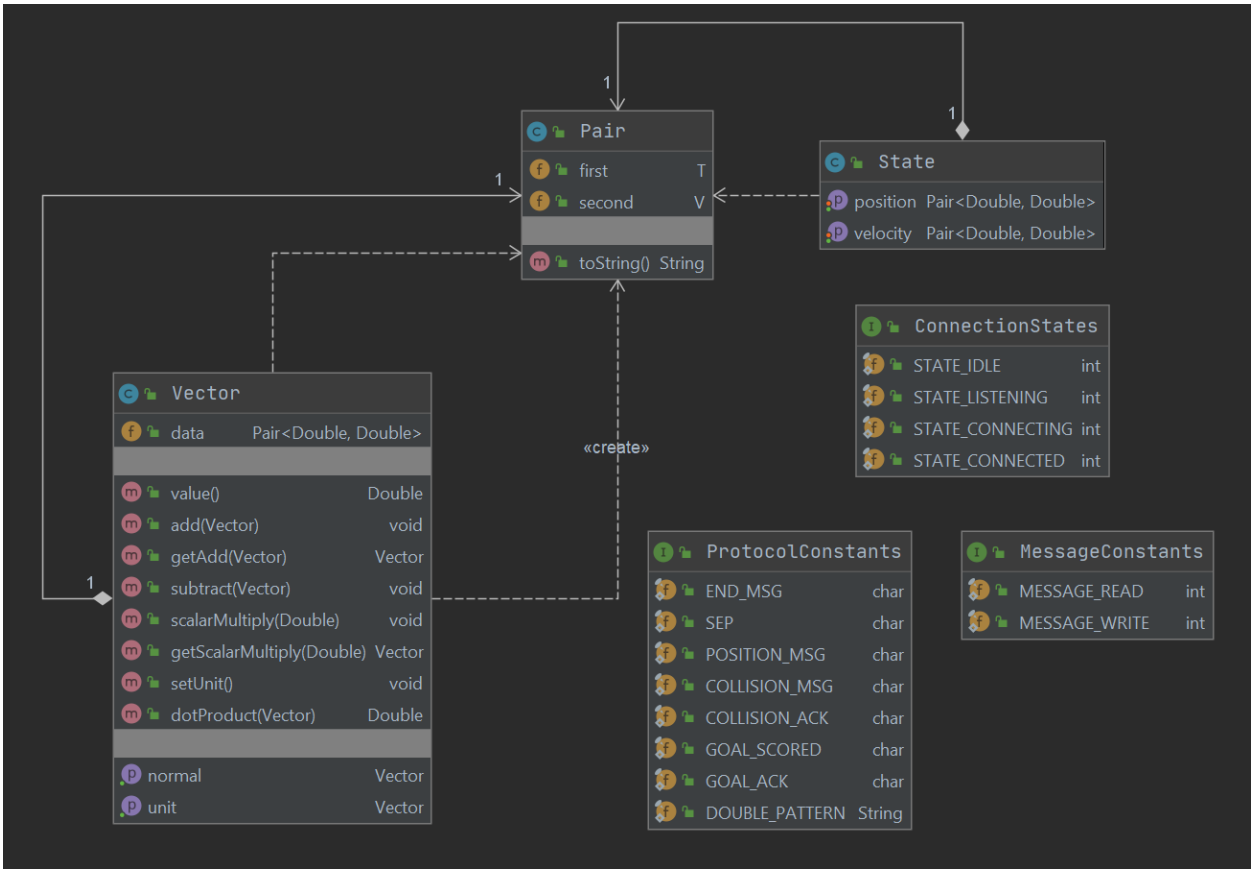


شکل ۴ – ساختار کلی فایل‌های پروژه

[10] Plugin
[11] Dependencies

- **بخش models:**

برای سادگی ابتدا به بخش models می‌پردازیم که در آن عناصر[12] اصلی بازی تعریف شده‌اند. همان‌طور که در شکل زیر قابل مشاهده است, این اجزا شامل مواردی مانند توپ بازی, زمین, دسته بازی, حالت[13] اجزای داخل صفحه بازی و غیره می‌شود.



شکل ۵ – ساختار بخش models پروژه

با توجه به شکل بالا مشخص می‌شود که ۶ کلاس به عنوان عناصر اصلی در این بخش وجود دارند.

[12] Objects
[13] State

❋ **رابط** *ConnectionState*[14] : این کلاس برای نگه‌داری وضعیت اتصال بین دو دستگاه استفاده می‌شود. وضعیت اتصال نیز طبق تعریف دارای ۴ حالت می‌باشد که با استفاده از integer مقداردهی می‌شوند. حالت‌ها نیز شامل موارد مقابل می‌شوند: IDLE یا همان ایستا, LISTENING یا همان آماده برقراری ارتباط, CONNECTING که نشان‌دهنده بودن در حین حالت متصل شدن بوده و در آخر هم CONNECTED که نشان‌دهنده متصل بودن است.

❋ **رابط** *ProtocolConstants*: این کلاس برای نگه‌داری کاراکترهایی که درون ارتباط بین دو دستگاه تعریف می‌شوند استفاده می‌شود. هر کدام از فیلدهای این کلاس شامل یک یا چند کاراکتر می‌باشد.

❋ **رابط** *MessageConstants*: این کلاس برای نگه‌داری دو حالت مربوط خواندن یا نوشتن می‌باشد که هر کدام با ۱ یا ۰ مشخص می‌شوند.

❋ **کلاس** *Pair* : این کلاس برای نگه‌داری جفت داده استفاده می‌شود. همان‌طور که انتظار می‌رود دو فیلد موجود در آن طول و عرض را مشخص می‌کنند. متغیرهای فیزیکی مانند سرعت و مکان از این کلاس بهره می‌برند.

❋ **کلاس** *State* : این کلاس شامل دو *Pair* می‌باشد. مورد اول برای نگه‌داری مکان و مورد دوم نیز برای نگه‌داری سرعت می‌باشد. همان‌طور که انتظار می‌رود, سرعت نیز مانند مکان در فضای این پروژه دارای دو بعد می‌باشد.

❋ **کلاس** *vector* :این کلاس به طور برای کار کردن با بردار ایجاد شده است. برای انجام محاسبات برخورد به دسته یا دیواره و انجام حرکت در جهت درست نیاز داریم که محاسبات برداری داشته باشیم. همان‌طور که مشاهده می‌شود, این کلاس به عنوان فیلد[15] از کلاس *Pair* استفاده می‌کند و به عنوان ویژگی نیز دارای یک *vector* به نام normal (که در واقع بردار عمود بر بردار فعلی می‌باشد) و *vector* دیگر به نام unit (که همان بردار یکه است) می‌باشد. متدهای این کلاس نیز به صورت زیر هستند:

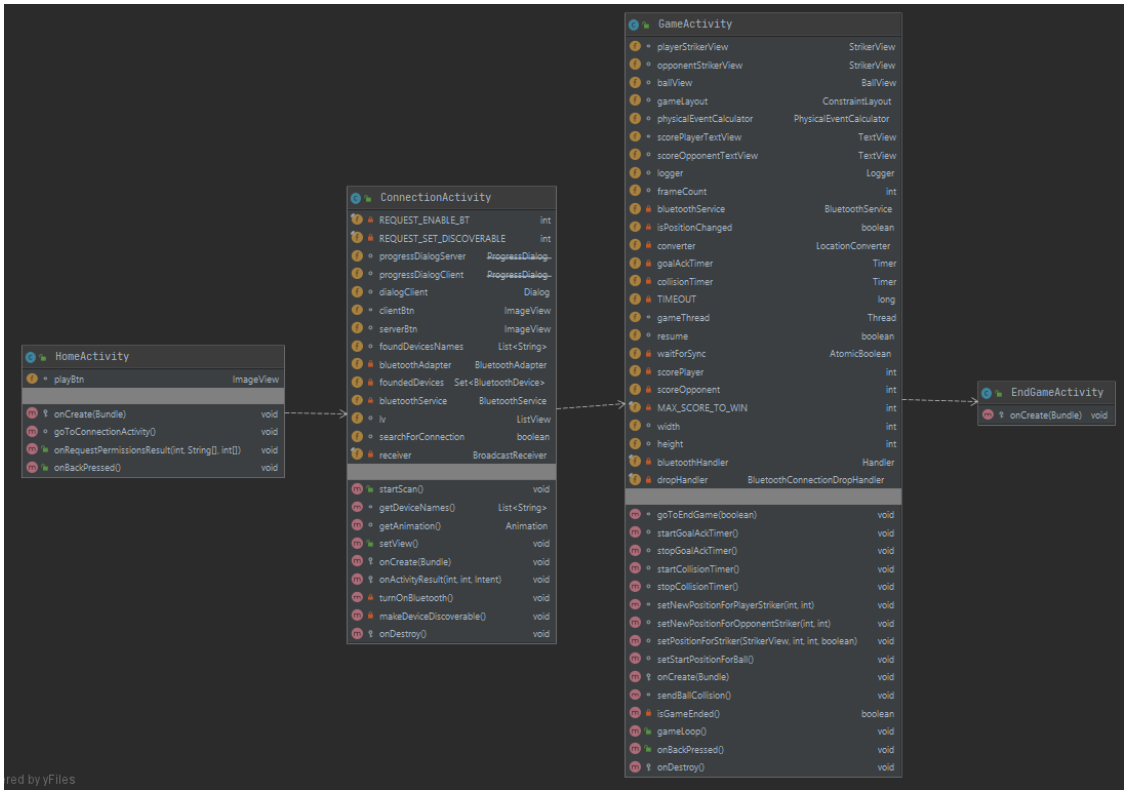
- متد value() : این متد اندازه بردار را محاسبه می‌کند.
- متد add(Vector) : این متد بردار ورودی را با بردار اصلی جمع می‌کند.
- متد getAdd(Vector) : این متد کاری مشابه متد بالا انجام داده و سپس مقدار جدید را برمی‌گرداند.
- متد subtract(Vector) : این متد بردار ورودی را از بردار اصلی کم می‌کند.
- متد scalarMultiply(Double) : این متد یک عدد ثابت در بردار ضرب می‌کند
- متد getScalarMultiply(Double) : این متد کاری مشابه متد بالا انجام داده و مقدار را برمی‌گرداند.
- متد setUnit() : برای مشخص کردن بردار یکه است.

[14] Interface
[15] Field

- متد doProduct(Vector) : این متد بردار ورودی را در بردار اصلی ضرب می‌کند.

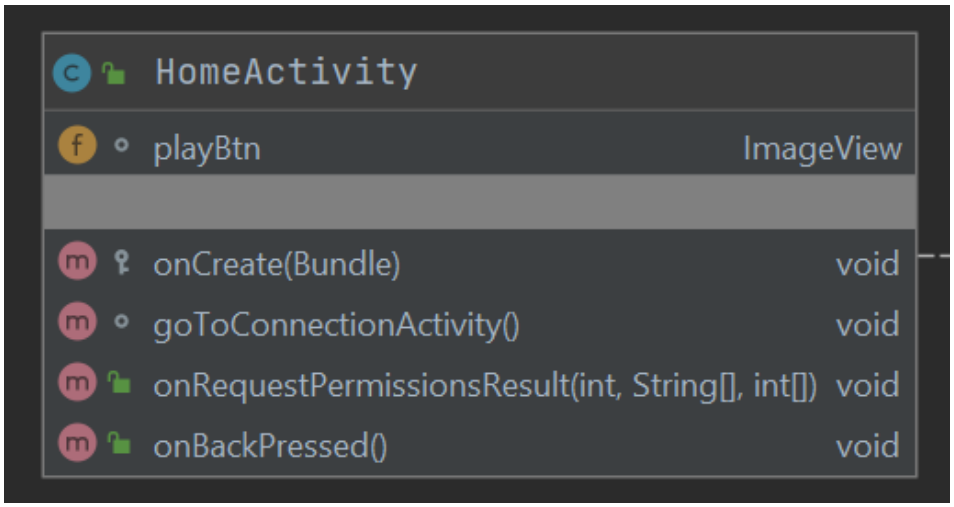
- **بخش activity:**

بخش اصلی بازی که activity های آن می‌باشد در این پوشه قرار دارند. همان‌طور که در شکل زیر مشخص است, ۴ عدد activity در ساختار این پروژه تعریف شده که توضیح هر کدام در ادامه گزارش قابل مشاهده است.



شکل ۶– ساختار بخش activity پروژه

❋ **کلاس** *HomeActivity* :



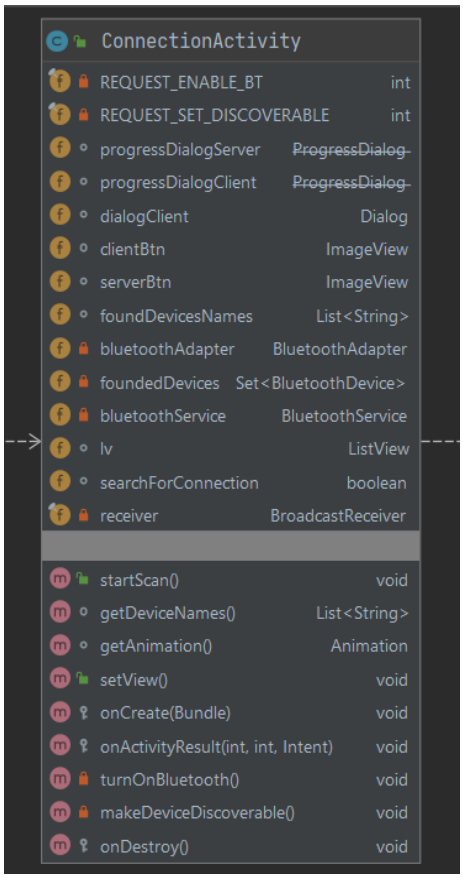
شکل ۷– ساختار کلاس HomeActivity **پروژه**

این کلاس نمایان‌گر Main Activity می‌باشد که بازی از آن شروع می‌شود. همان‌طور که مشخص است یک دکمه تحت عنوان PlayBtn در این قسمت تعریف شده و متدهای این کلاس نیز به صورت زیر هستند:

- متد onCreate(Bundle) : برای شروع بازی و کار کردن با دکمه شروع استفاده می‌شود.
- متد goToConnectionActivity() : این متد برای رفتن به Activity بعدی می‌باشد که توضیحات آن در ادامه داده شده است.
- متد onRequestPermissionResult(int, String, int) : این متد برای انجام دادن بحث دریافت اجازه از کاربر برای دادن دسترسی به ماژول‌های مورد نیاز برنامه و کاری که در قبال هر حالت باید انجام شود طراحی شده است.
- متد onBackPressed() : این متد برای این است که در صورتی که Logger در یک ریسه [۱۶]جدا در حال کار کردن است با این متد آن ریسه متوقف می‌شود.

[16] Thread

❋ **کلاس** *ConnectionActivity* :



شکل ۸ – ساختار کلاس ConnectionActivity **پروژه**

این کلاس برای انجام دادن کارهای مربوط به جستجو, یافتن و متصل شدن به دستگاه دیگر از طریق ماژول بلوتوث کاربرد دارد. بر خلاف کلاس‌هایی که تا کنون بررسی شدند این کلاس دارای تعداد بیشتری فیلد می‌باشد لذا هر کدام را به طور خلاصه توضیح می‌دهیم:

- فیلد REQUEST_ENABLE_BT : مربوط به فعال بودن ماژول بلوتوث دستگاه
- فیلد REQUEST_SET_DISCOVERABLE : برای روشن کردن حالت Discoverable ماژول بلوتوث
- فیلد progressDialogServer : هنگام متصل شدن به دستگاه دیگر زمانی لازم است که طرف سرور منتظر مانده تا اتصال انجام شود. این زمان با یک ProgressDialog در برنامه نشان داده می‌شود.
- فیلد progressDialogClient : این فیلد مشابه فیلد بالا بوده با این تفاوت که برای سمت Client است.
- فیلد dialogClient : مربوط به صفحه‌ای است که برای بخش انتظار متصل شدن کاربرد دارد.
- فیلد clientBtn : دکمه مربوط به اتصال برای سمت Client
- فیلد serverBtn : دکمه مربوط به اتصال برای سمت Server

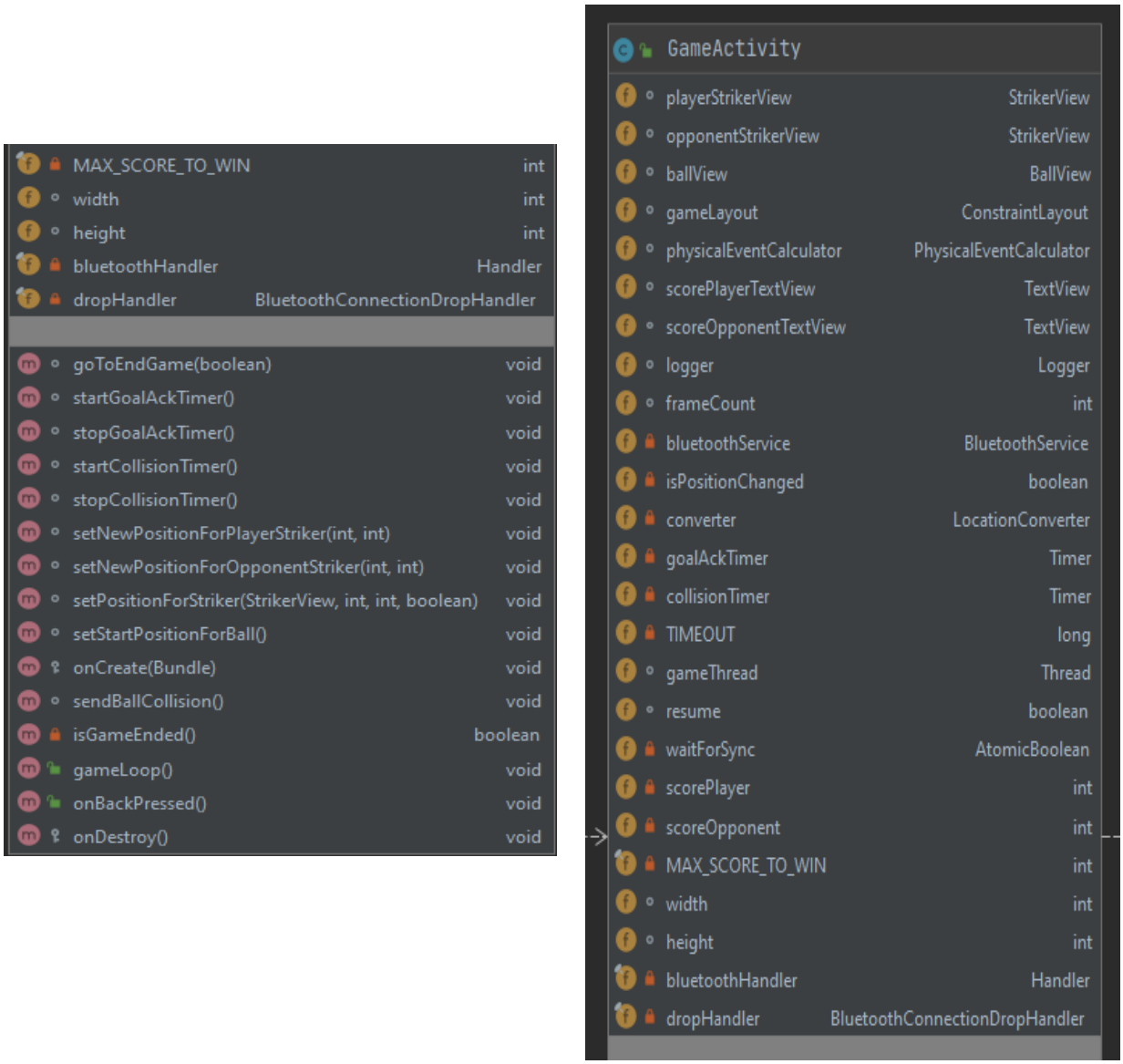
- فیلد foundDevicesNames : یک لیست که نام دستگاه‌های یافت شده را نگه‌داری می‌کند.
- فیلد bluetoothAdapter : آداپتور[۱۷] برای کار کردن با ماژول بلوتوث
- فیلد foundDevices : یک لیست غیرتکراری از BluetoothDevice هایی که یافت شده‌اند.
- فیلد bluetoothService : سرویس برای کار کردن با ماژول بلوتوث
- فیلد lv : این فیلد یک ListView می‌باشد که در آخرین نسخه پروژه بدون استفاده است.
- فیلد searchForConnection : از جنس Boolean بوده و به دنبال اتصال بودن را مشخص می‌کند.
- فیلد receiver : دریافت‌کننده برای کار کردن با ماژول بلوتوث

همچنین متدهای مهمی (جز متدهایی که برای مواردی مثل get و یا set استفاده می‌شوند) که در این کلاس مورد استفاده قرار گرفته‌اند نیز به شرح زیر می‌باشند:

- متد startScan() :این متد برای شروع کردن فرایند گشتن به دنبال دستگاه‌های دیگر است.
- متد getAnimation() : برای ساختن انمیشن اولیه که مانند تپش قلب در فرایند اتصال نشان داده می‌شود.
- متد onCreate(Bundle) : برای آغاز به کار کلاس و روشن کردن ماژول بلوتوث و غیره.
- متد onActivityResult(int, int, intent) : این متد برای تعریف کردن عکس‌العمل‌هایی که نسبت به حالات مختلفی که از ماژول بلوتوث مانند روشن شدن و Discoverable شدن داریم استفاده می‌شود.
- متد turnOnBluetooth() : برای روشن کردن ماژول بلوتوث دستگاه کاربرد دارد.
- متد makeDeviceDiscoverable() : برای روشن کردن تنظیم Discoverable بودن دستگاه کاربرد دارد.
- متد onDestroy() : برای بحث‌های پایانی مانند غیرفعال کردن receiver و خاموش کردن فعالیت Discovery استفاده می‌شود.

[17] Adapter

❋ کلاس *GameActivity* :



شکل ۹ – ساختار کلاس GameActivity **پروژه**

این کلاس در واقع کلاس اصلی بازی می‌باشد و برای انجام دادن بازی بین دو طرف کاربرد دارد. از جمله وظایف آن می‌توان به مواردی مانند محاسبه مکان دسته‌ها و توپ, نگه داشتن وضعیت بازی در هر لحظه و پیاده‌سازی منطق حاکم بر بازی مانند سیستم امتیازدهی و برخوردها اشاره کرد. ابتدا فیلدهای این کلاس را شرح می‌دهیم:

- فیلد playerStrikerView : که برای نشان دادن دسته بازیکن کنونی استفاده می‌شود.
- فیلد opponentStrikerView : مانند فیلد بالا ولی برای دسته بازیکن حریف کاربرد دارد.
- فیلد ballView : برای نشان دادن توپ بازی استفاده می‌شود.

- فیلد gameLayout : لایه اصلی مربوط به بازی که از جنس View می‌باشد.
- فیلد physicalEventCalculator : یک کلاس جدا است که در قسمت‌های بعدی توضیح آن موجود بوده و وظیفه اصلی آن پیاده‌سازی فیزیک اصلی حاکم بر بازی است.
- فیلد scorePlayerTextView : برای نشان دادن امتیاز بازیکن کنونی است.
- فیلد scoreOpponentTextView : برای نشان دادن امتیاز بازیکن حریف است.
- فیلد logger : یک کلاس جدا است که در قسمت‌های بعدی توضیح آن موجود بوده و وظیفه اصلی آن log کردن اتفاقات بازی و موارد مربوط به اتصال دو دستگاه است.
- فیلد frameCount : نشان‌دهنده تعداد فریم‌هایی است که از شروع بازی جلو رفته است.
- فیلد bluetoothService : توضیح آن در قسمت‌های قبل موجود است.
- فیلد isPositionChanged : برای بررسی اینکه آیا مکان دسته بازیکن عوض شده یا خیر استفاده می‌شود.
- فیلد converter : یک کلاس جدا است که در قسمت‌های بعدی توضیح آن موجود بوده و وظیفه اصلی آن تبدیل کردن اطلاعات مکانی دریافتی از دستگاه دیگر به مقیاس درست منطبق بر دستگاه فعلی است.
- فیلد goalAckTimer : یک فیلد از جنس Timer بوده که زمان در دسترس برای دریافت تاییدیه[18] ثبت شدن گل می‌باشد.
- فیلد collisionTimer : این فیلد نیز مشابه بالا بوده ولی برای برخورد استفاده می‌شود.
- فیلد TIMEOUT : این فیلد در Timer هایی که در بالا توضیح داده شد استفاده می‌شود و یک عدد است.
- فیلد gameThread : ریسه اصلی مربوط به بازی می‌باشد.
- فیلد resume : این فیلد مشخص می‌کند که حلقه بازی ادامه پیدا کرده یا خاتمه یابد.
- فیلد waitForSync : این فیلد برای هماهنگ‌سازی بازی از دید دو دو بازیکن بعد از ثبت گل استفاده می‌شود.
- فیلد scorePlayer : امتیاز بازیکن کنونی است.
- فیلد scoreOpponent : امتیاز بازیکن حریف است.
- فیلد MAX_SCORE_TO_WIN : امتیازی نهایی برای پایان بازی را مشخص می‌کند.
- فیلد width : برای نگه‌داری عرض زمین استفاده می‌شود.
- فیلد height : برای نگه‌داری طول زمین استفاده می‌شود.

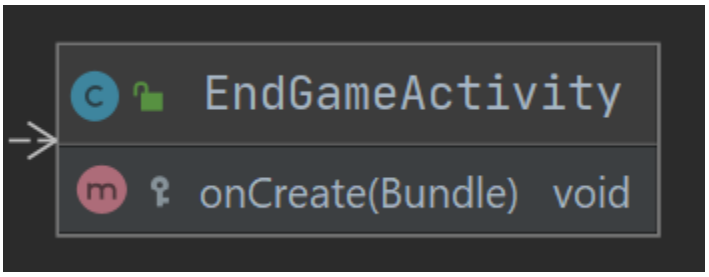
[18] Acknowledgment

- فیلد bluetoothHandler : یک فیلد از جنس Handler می‌باشد که پیام‌هایی که از طریق ماژول بلوتوث می‌رسد را دریافت کرده تا تصمیم‌گیری درست در همین کلاس برای کارهای بعدی انجام شود.
- فیلد dropHandler : برای تشخیص دادن از دست رفتن اتصال بلوتوث می‌باشد. این فیلد برای در کنار فرایند تشخیص از دست رفتن ارتباط که در پیاده‌سازی اصلی ماژول بلوتوث می‌باشد بوده و روش کار آن هم این است که در صورتی که دو پیام با مقدار ۱- دریافت شده آنگاه یک پیام خالی با شماره ۰ توسط Handler ارسال شده و این *GameActivity* نیز از این طریق متوجه از دست رفتن اتصال شده و بازی را به صفحه پایانی منتقل می‌کند.

متدهای مهم این کلاس نیز به شرح زیر می‌باشند:

- متد goToEndGame(Boolean) : برای رفتن به پایان بازی و فرایندهای خاتمه کاربرد دارد.
- متد start/stopGoalAckTimer() : شروع و یا پایان دادن به زمان‌سنج مربوط به تاییدیه ثبت گل.
- متد start/stopCollisionTimer() : شروع و یا پایان دادن به زمان‌سنج مربوط به برخورد.
- متد onCreate(Bundle) : برای آغاز به کار کلاس کاربرد دارد.
- متد sendBallCollision() : از طریق ماژول بلوتوث برخورد را به اطلاع دستگاه دیگر می‌رساند.
- متد isGameEnded() : متدی که برای برگرداندن وضعیت خاتمه بازی استفاده می‌شود.
- متد gameLoop() : حلقه اصلی بازی که جریان بازی در آن اتفاق می‌افتد.
- متد onBackPressed() : این متد فیلد resume را False کرده و ترد بازی تمام می‌شود.
- متد onDestroy() : برای بحث‌های پایان دادن به کلاس و قطع کردن اتصال سرویس بلوتوث کاربرد دارد.

❋ **کلاس** *EndGameActivity* :

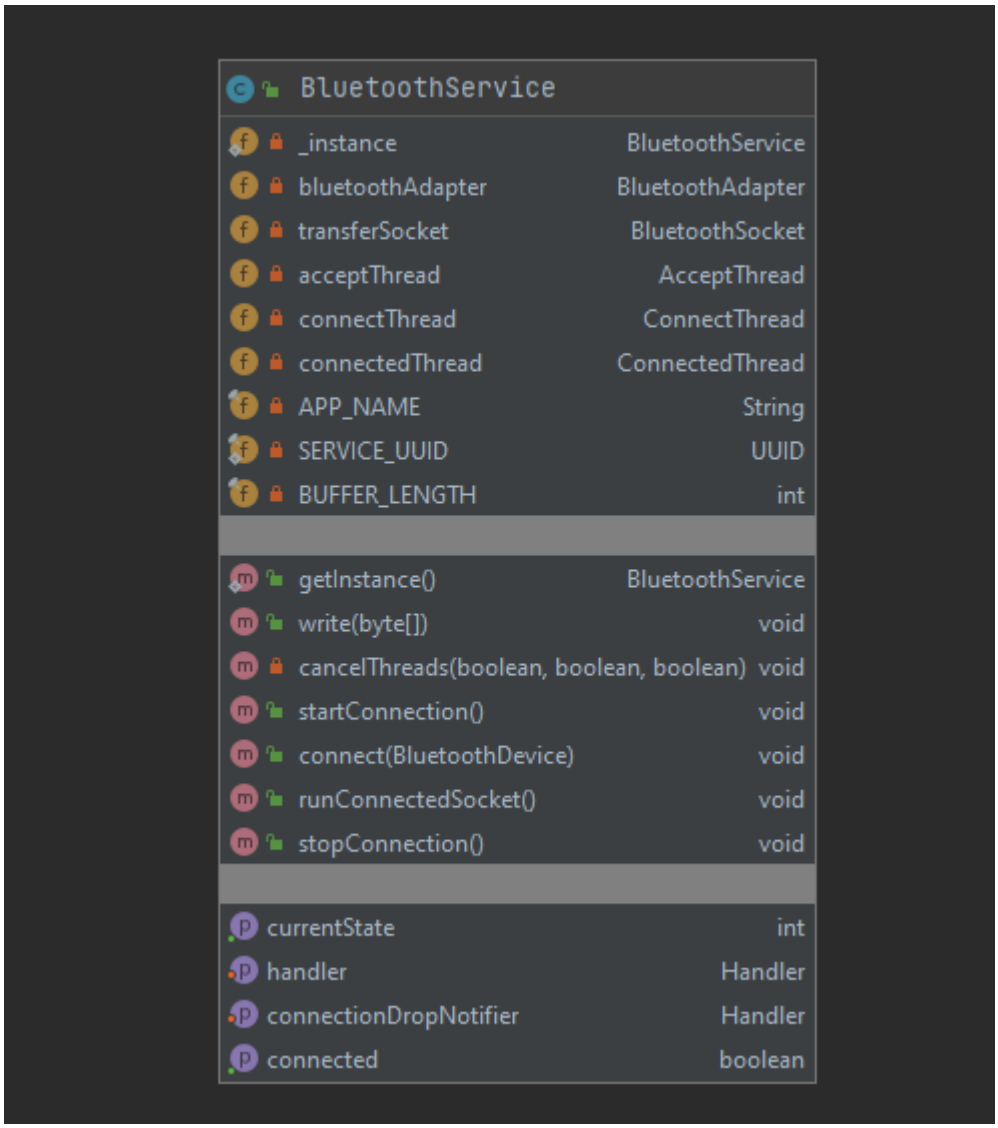


شکل ۱۰– ساختار کلاس EndGameActivity **پروژه**

این کلاس برای پایان بازی و نشان دادن صفحه نهایی نتیجه استفاده می‌شود. یک متد onCreate(Bundle) داشته که کارهای کل کلاس نیز توسط همین متد انجام می‌شود.

- **بخش services:**

این بخش صرفا شامل یک کلاس می‌شود که آن کلاس هم برای انجام دادن کارهای ماژول بلوتوث مانند متصل شدن و رد و بدل کردن داده می‌باشد. داده‌های دریافتی توسط این بخش در بخش‌های دیگر برای پیاده‌سازی فرایند بازی استفاده می‌شوند.



شکل ۱۱– ساختار بخش services **پروژه**

* **کلاس** *BluetoothService* : توضیحات این کلاس همان مواردی است که برای کل بخش services در بالا مطرح شد. در این کلاس به تعداد ۴ عدد ویژگی نیز داریم که دو Handler ای که در بخش activity توضیح داده شدند در این بخش حاضر هستند. دو ویژگی دیگر نیز که connected و currentState هستند برای نگه‌داری وضعیت استفاده می‌شوند. فیلدهای این کلاس نیز به شرح زیر هستند:

- فیلد _instance : سرویس برای کار کردن با ماژول بلوتوث
- فیلد bluetoothAdapter : آداپتور برای کار کردن با ماژول بلوتوث
- فیلد transferSocket : یک سوکت از جنس BluetoothSocket برای رد و بدل کردن داده‌ها
- فیلد acceptThread : یک ریسه برای قبول کردن اتصال قبل از فاز متصل شدن.

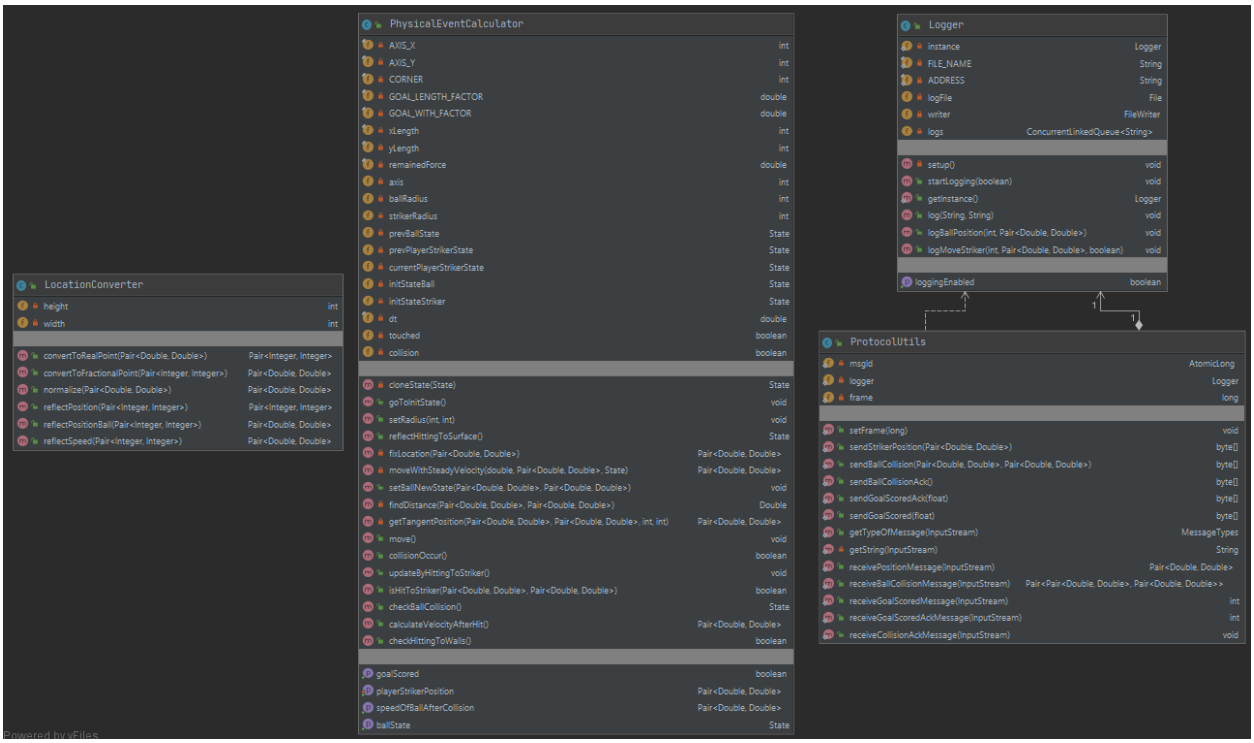
- فیلد connectThread : یک ریسه برای متصل شدن به دستگاه دیگر از طریق ماژول بلوتوث.
- فیلد connectedThread : یک ریسه برای زمانی که اتصال بین دو دستگاه برقرار است.
- فیلد APP_NAME : نام اپلیکیشن را مشخص می‌کند که در این پروژه AirHockey است.
- فیلد SERVICE_UUID : شناسه‌ای یکتا برای مشخص کردن اپلیکیشنی که به آن متصل می‌شویم.
- فیلد BUFFER_LENGTH : این فیلد سایز بافر را مشخص می‌کند که در اینجا ۱۰۲۴ است.

متدهای مهم این کلاس نیز به شرح زیر می‌باشند:

- متد write(byte[]) : برای نوشتن داده به سوکت اصلی ارتباط
- متد cancelThreads(Boolean, Boolean, Boolean) : این متد برای پایان دادن به یک یا چند تا از ریسه‌های بازی استفاده می‌شود.
- متد startConnection() : متد برای آغاز کردن ریسه acceptThread می‌باشد
- متد connect(BluetoothDevice) : متد برای انجام موارد مربوط به فرایندی که از یافتن دستگاه مقابل تا متصل شدن سوکت‌ها ادامه می‌یابد می‌باشد.
- متد runConnectedSocket() : متد برای اجرای سوکت اصلی ارتباط که در ریسه connectedThread اجرا می‌شود.
- متد stopConnection() : این متد برای بستن هر سه ریسه‌ای که در این بخش معرفی شد کاربرد دارد.

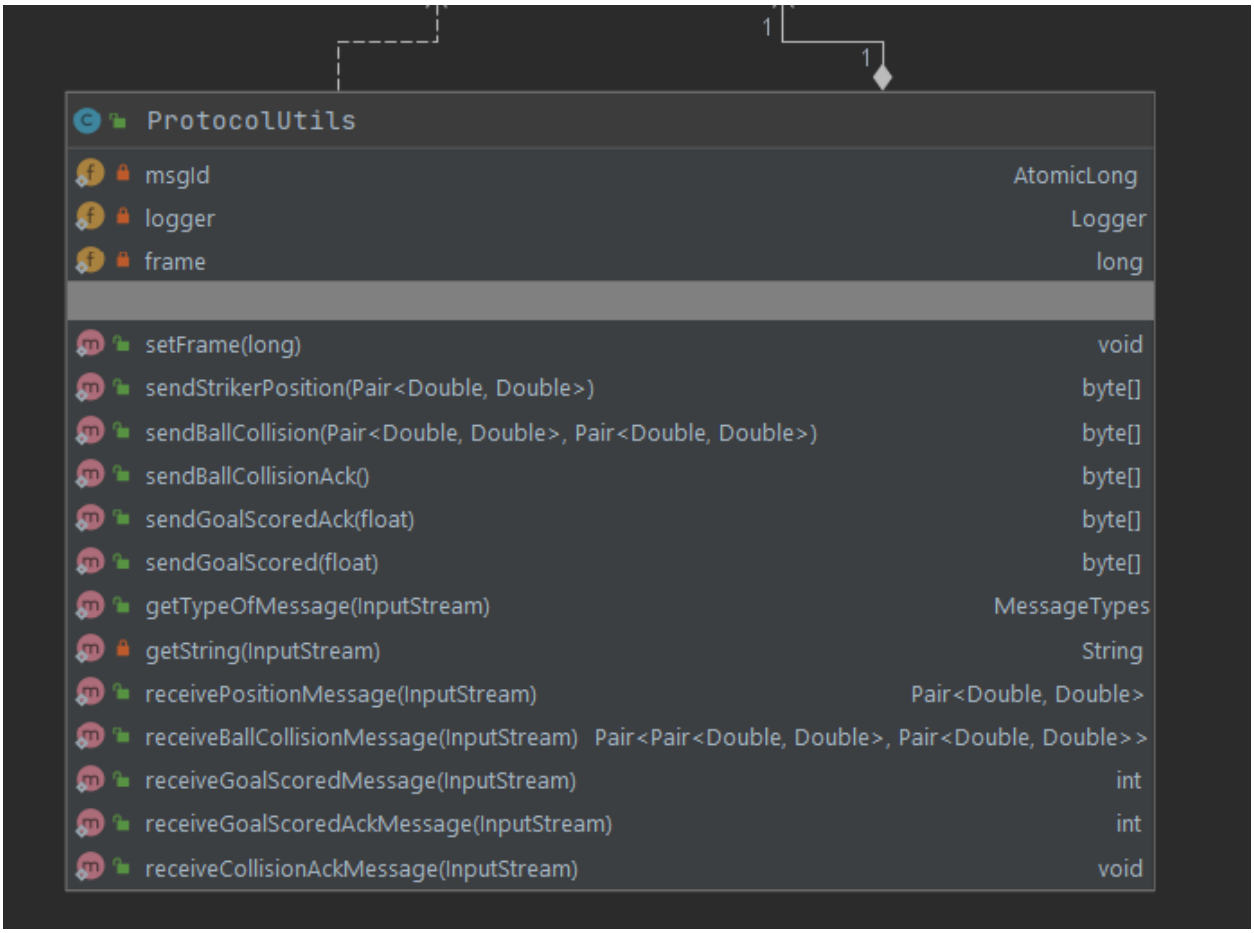
- **بخش utils:**

این بخش شامل ۴ کلاس شده که وظایف مختلفی را دارا هستند. شکل زیر ساختار کلی این کلاس‌ها را نشان می‌دهد:



شکل ۱۲ – ساختار بخش utils **پروژه**

❋ **کلاس** *ProtocolUtils* :

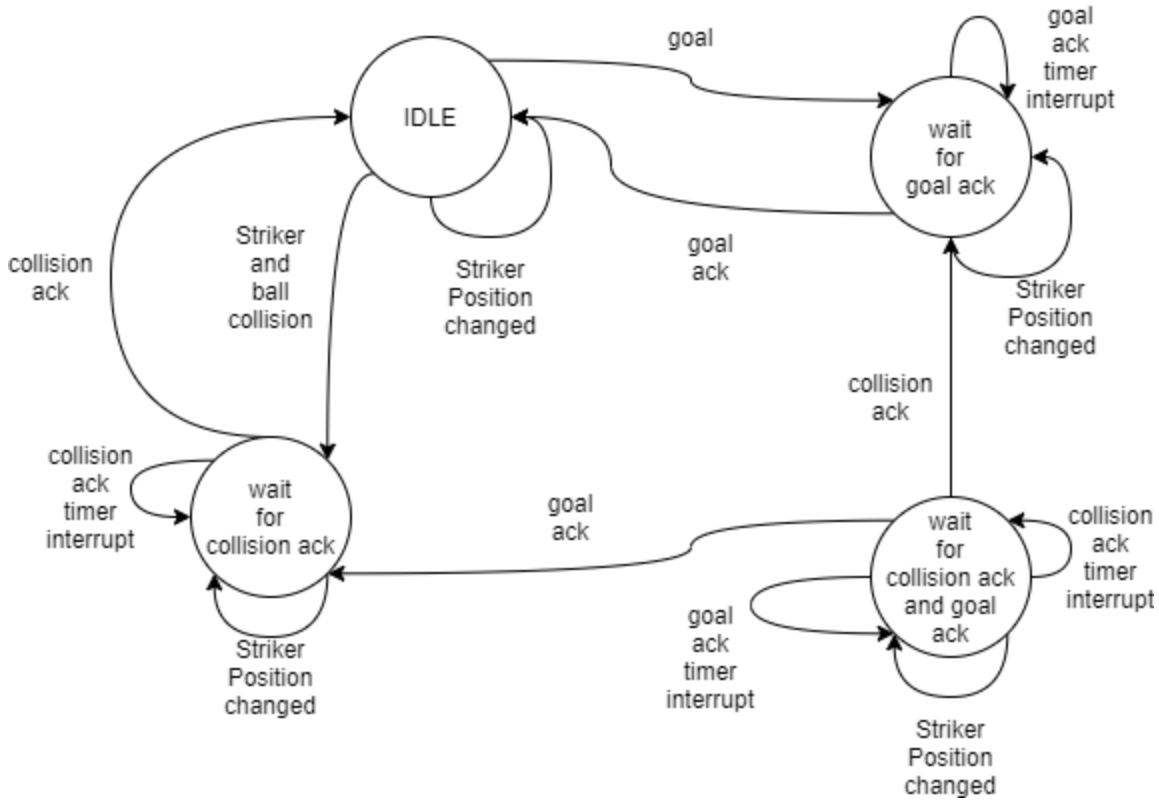


شکل ۱۳ – ساختار کلاس ProtocolUtils **پروژه**

این کلاس برای انجام دادن کارهای مربوط به پروتکل ارتباطی بین دو دستگاه کاربرد دارد. فیلدهای آن شامل موارد

- فیلد msgId : آیدی مختص به پیام
- فیلد logger : کلاس Logger که در کنار همین کلاس تعریف شده است.
- فیلد frame : شماره فریم کنونی

برای توضیح نحوه کار کردن این کلاس به‌جای دادن توضیحات خلاصه مانند قبل, نمودار زیر تولید شده است تا فهم بهتری از فرایند ارسال پیام‌های مختلفی مانند تاییدیه, ثبت گل و یا برخورد بتوان داشت:

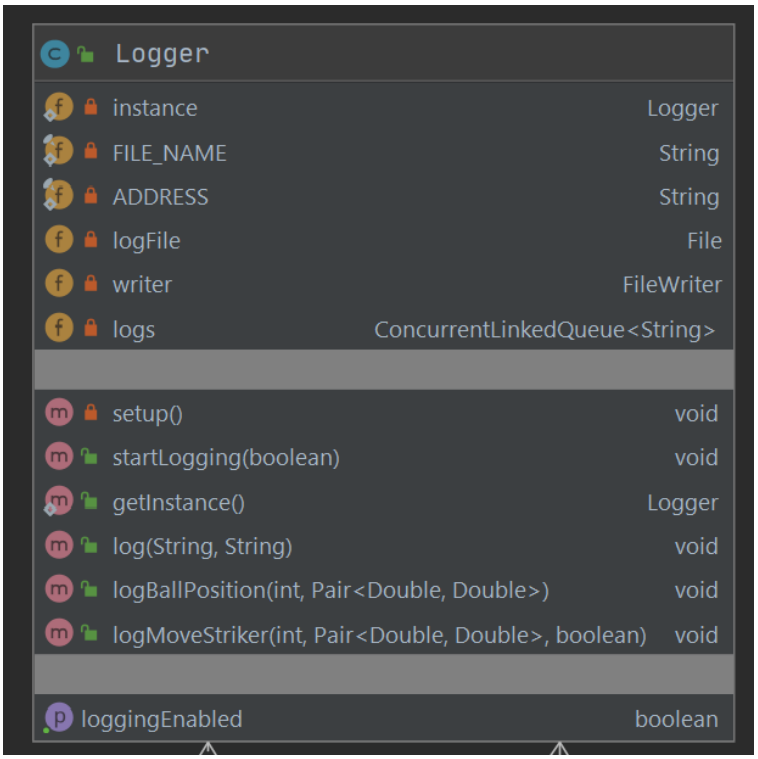


شکل ۱۴ – نمودار حالت پروتکل‌های ارتباطی پروژه

همان‌طور که در شکل بالا نیز قابل مشاهده است, ۴ حالت کلی در این بخش وجود دارد که شامل موارد زیر می‌شود:

- حالت IDLE
- حالت wait for goal ack
- حالت wait for collision ack
- حالت wait for collision ack and goal ack

❋ **کلاس** *Logger* :



شکل ۱۵ – ساختار کلاس Logger **پروژه**

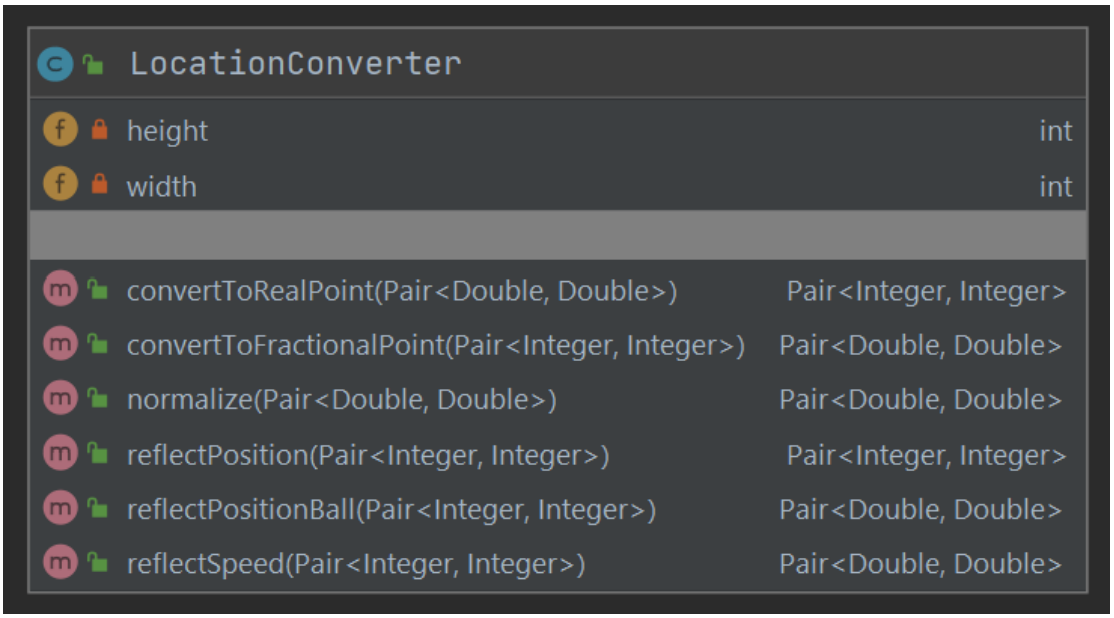
این کلاس برای انجام دادن عمل logging استفاده می‌شود. همان‌طور که در تصویر بالا نیز قابل مشاهده است, این کلاس شامل فیلدهای زیر می‌شود:

- فیلد instance : این فیلد خود کلاس Logger است.
- فیلد FILE_NAME : این فیلد برای مشخص کردن نام فایلی مقصد استفاده می‌شود.
- فیلد ADDRESS : این فیلد برای مشخص کردن آدرس فایل مذکور استفاده می‌شود.
- فیلد logFile : این فیلد از جنس File می‌باشد و مشخص‌کننده فایل مذکور است.
- فیلد writer : این فیلد از جنس FileWriter بوده و برای نوشتن در فایل مذکور استفاده می‌شود.
- فیلد logs : این فیلد حاوی خود پیغام‌های log ای است که می‌خواهیم ذخیره کنیم.

متدهای مهم این کلاس نیز به شرح زیر هستند:

- متد setup() : این متد برای انجام کارهای اولیه مانند ساختن فایل در آدرس مشخص کاربرد دارد.
- متد startLogging(boolean) : این متد صرفا برای فعال‌سازی نوشتن log ها استفاده میشود.
- متدهای log/logBallPosition/logMoveStriker : این سه متد برای انجام دادن خود بحث logging کاربرد دارند.

❋ **کلاس** *LocationConverter* :

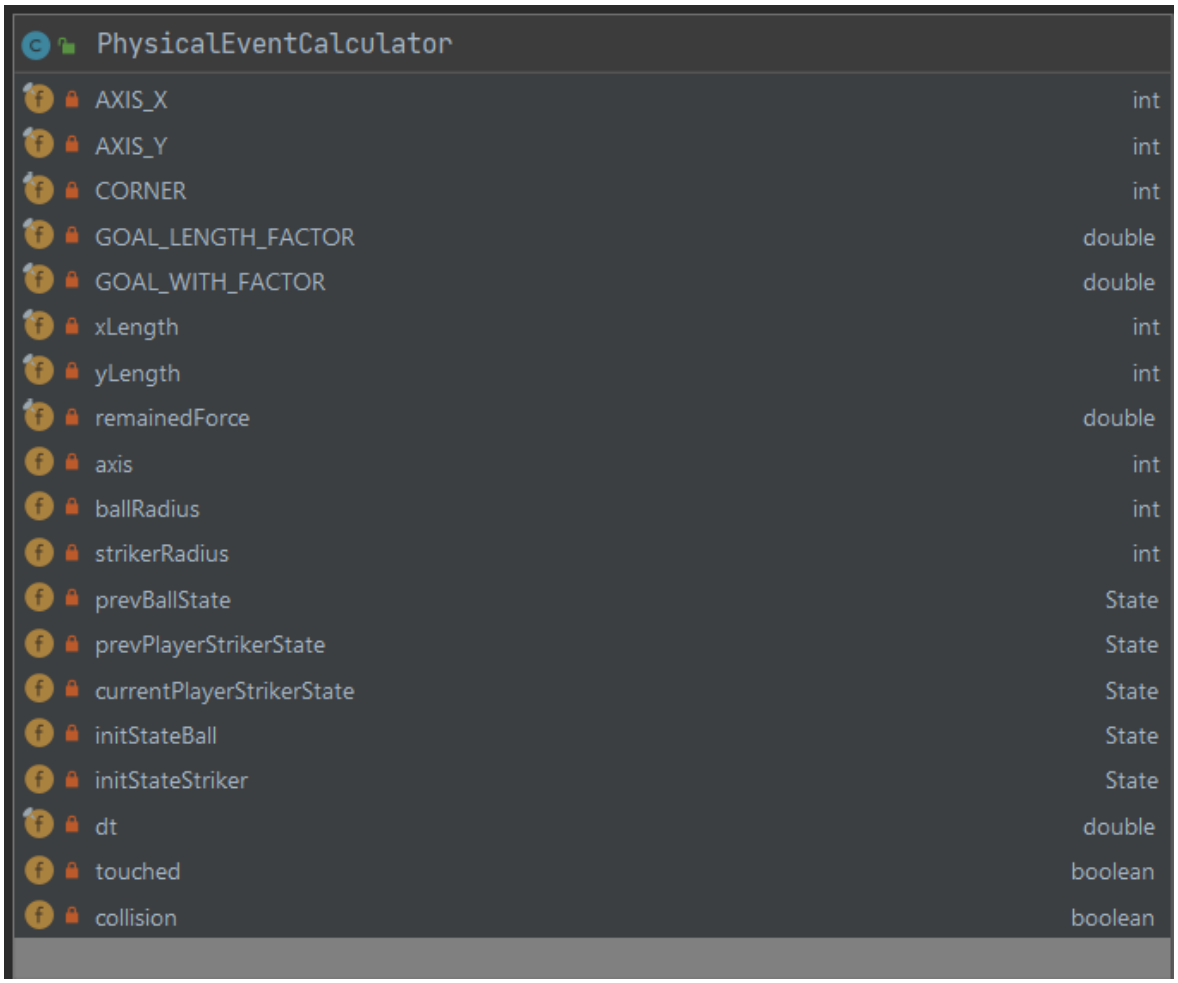


شکل ۱۶ – ساختار کلاس LocationConverter **پروژه**

این کلاس برای انجام تبدیلاتی که برای هر دستگاه لازم است کاربرد دارد. بدلیل اینکه ممکن است سایز صفحه هر دستگاه از دیگری متفاوت باشد, این نیاز وجود دارد که همه حرکت‌های صفحه در مقیاس درست همان صفحه محاسبه و نشان داده شوند. فیلدهای این کلاس شامل width و height بوده که همان طول و عرض صفحه می‌باشد. متدهای این کلاس نیز سه بخش دارند:

- متدهای convert() : به منظور تبدیل کردن داده‌های دریافت شده به مقیاس مناسب صفحه
- متد normalize() : برای نرمال کردن (تبدیل integer به double) از خروجی متدهای قبلی
- متدهای reflect() : به دلیل اینکه هر زمین هر بازیکن قرینه بازیکن حریف است, لازم است که داده‌هایی که از طریق ماژول بلوتوث از حریف دریافت می‌شود به نوعی قرینه شده تا به درستی محاسبه شوند.

❋ **کلاس** *PhysicalEventCalculator* :

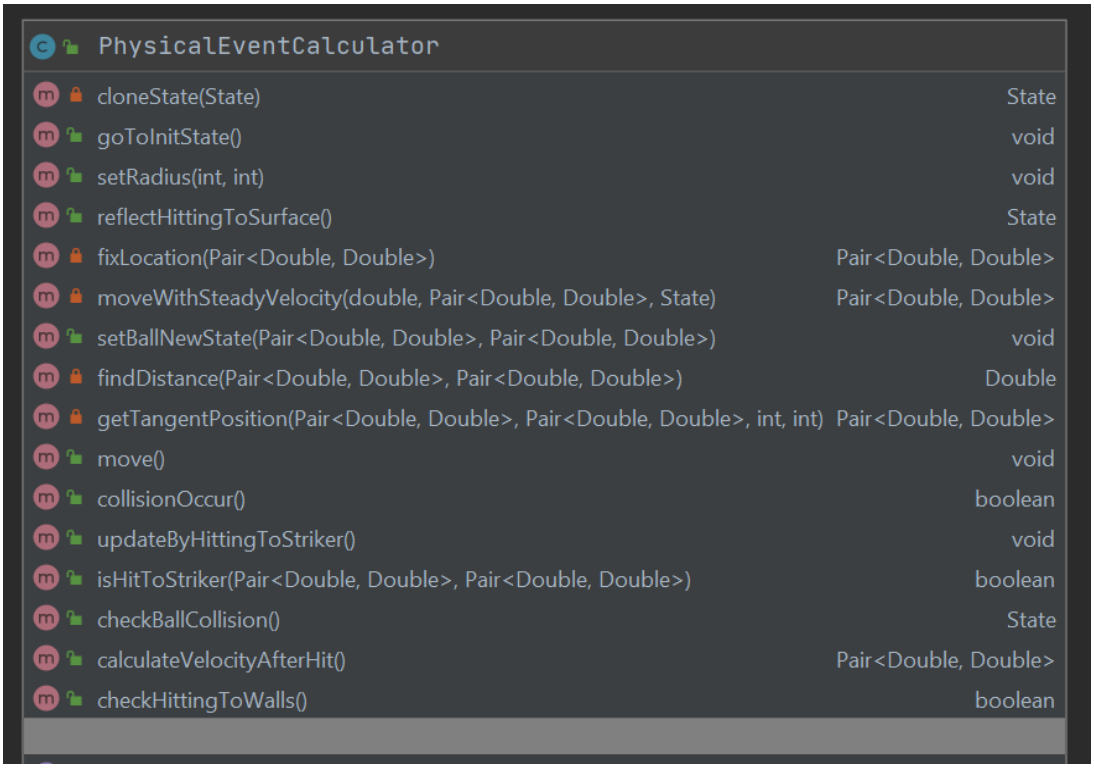


شکل ۱۷ – ساختار فیلدهای کلاس PhysicalEventCalculator **پروژه**

این کلاس برای انجام بحث محاسبات و پیاده‌سازی منطق اصلی فیزیک بازی استفاده می‌شود. در شکل بالا می‌توان فیلدهای موجود در این کلاس را مشاهده کرد. برخی از موارد مهم این فیلدها که در نگاه اول قابل تشخیص نمی‌باشند به شرح زیر هستند:

- فیلد axis : مشخص کردن اینکه محور مورد نظر محور x یا y است.
- فیلدهای prevState : برای محاسبه State کنونی همواره از State قبلی استفاده می‌کنیم.
- فیلد dt : مشخص‌کننده زمان سپری شده بین فریم کنونی تا فریم بعدی است.
- فیلد touched : این فیلد مشخص می‌کند که آیا توپ دسته را لمس کرده است یا خیر. اولین باری که توپ به دسته خورده True می‌شود.

- فیلد collision : این فیلد مشخص می‌کند که آیا توپ و دسته برخورد داشته‌اند یا خیر. این فیلد بر خلاف فیلد قبلی به کدامین دفعه کاری نداشته و صرفا برای برخورد در هر لحظه استفاده می‌شود.



شکل ۱۸ – ساختار متدهای کلاس PhysicalEventCalculator **پروژه**

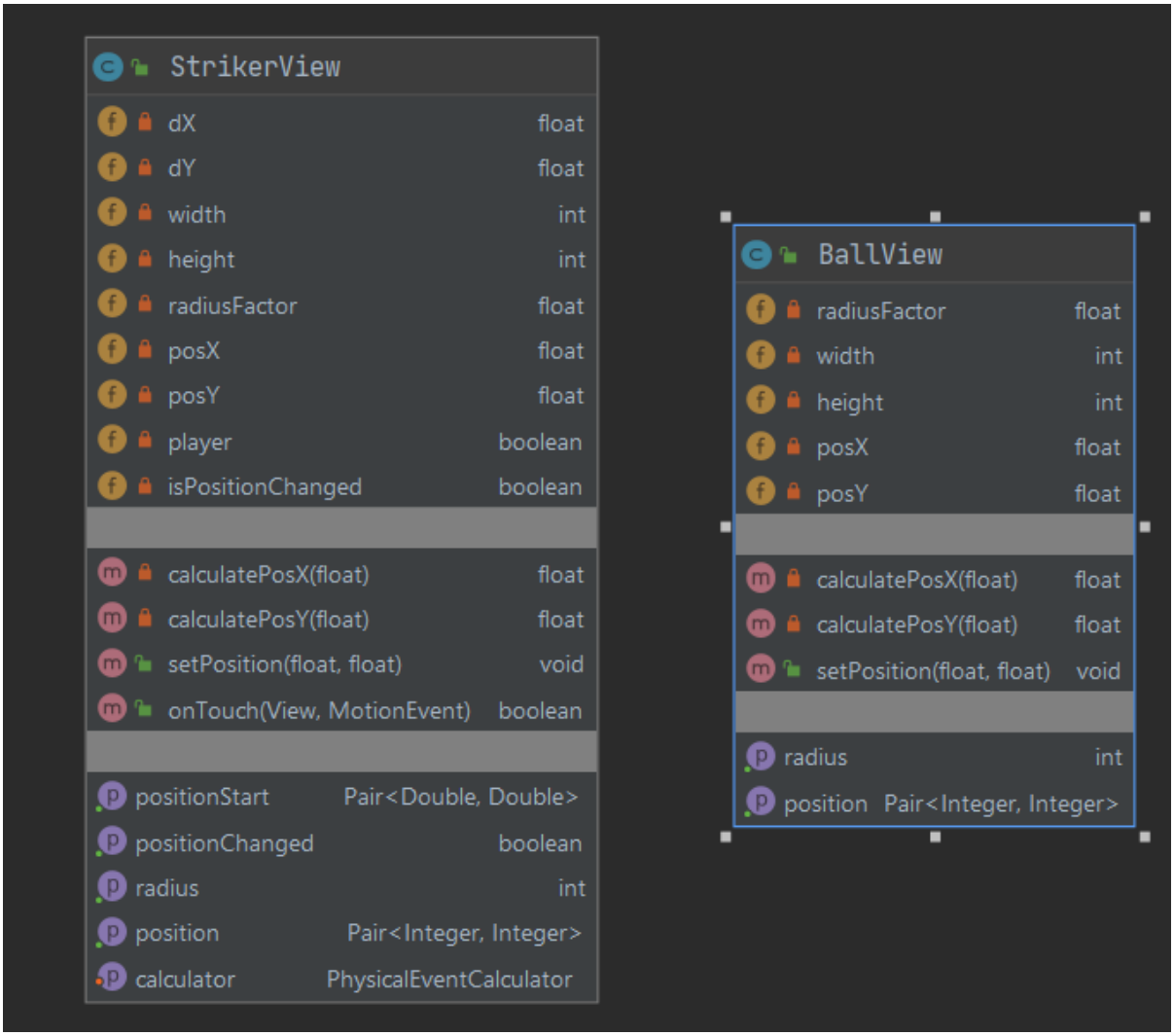
شرح مهم‌ترین متدهای این کلاس نیز به صورت زیر است:

- متد cloneState(State) : این متد برای ساختن یک کپی از State ورودی به کار می‌رود.
- متد goToInitState() :این متد برای رفتن به حالت اولیه کاربرد دارد.
- متد reflectHittingToSurface() : این متد برای مشخص کردن State توپ هنگامی که به جداره‌های صفحه برخورد می‌کند مورد استفاده قرار می‌گیرد.
- متد fixLocation(…) : این متد برای کنترل خارج نشدن توپ از صفحه هنگام برخورد با دیواره‌های زمین مورد استفاده قرار می‌گیرد.
- متد moveWithSteadyVelocity(…) : این متد برای مدل کردن حرکت به سرعت ثابت برای اجزای بازی مورد استفاده قرار می‌گیرد.
- متد move() : متد که State بعدی کل بازی شامل توپ و دسته رو تعیین کرده و رو جابه‌جایی را هم انجام می‌دهد.

- متد checkBallCollision() : این متد State بعدی توپ را هنگامی که توپ به دسته برخورد کرده محاسبه می‌کند. (علاوه بر تشخیص برخورد)
- متد checkHittingToWalls(): متد برای بررسی این که آیا توپ به دیواره برخورد کرده یا خیر.

- **بخش view:**

این بخش شامل ۲ کلاس شده که یکی برای نمایش دادن توپ و دیگری برای نمایش دسته بکار می‌رود. ساختار کلی این دو کلاس در شکل زیر قابل مشاهده است:



شکل ۱۹– ساختار بخش view پروژه

به دلیل سادگی فیلدها و متدهای دو کلاس بالا نیازی به مطرح کردن توضیحات هر بخش در گزارش نبوده و کد هر بخش در کنار نام متد نمایان‌گر کاربرد هر بخش می‌باشد. صرفا لازم به ذکر است که StrikerView به عنوان ویژگی دارای موارد مانند positionStart به معنی مختصات شروع, position به معنای مختصات کنونی و

calculator است که یک کلاس از جنس PhysicalEventCalculator بوده که توضیحات آن در قسمت‌های قبلی داده شده است.

- **تغییرات اعمال شده(بیشتر برای پروژه‌های سیستم‌عامل محور)**

به دلیل اینکه این در این پروژه با استفاده از تنظیمات معمول سیستم‌عامل اندروید امکان کار کردن با ماژول بلوتوث, انجام محاسبات و نمایش اجزای بازی با کیفیت‌های در نظر گرفته‌شده وجود داشت, از انجام تغییرات در سطح سیستم‌عامل خودداری به‌عمل آمد.

۷- تست عملکرد

- **طرح تست**

برای تست کردن عملکرد برنامه علاوه بر انجام بازی روی دو دستگاه و انجام مشاهدات از دید دو بازیکن, همان‌طور که در بخش پیاده‌سازی برنامه نیز توضیح داده شد, یک کلاس Logger در نظر گرفتیم تا هنگام اجرای بازی اطلاعات مختلفی که از کیفیت ارتباط بین دو دستگاه می‌توان داشت را ذخیره کند. سپس با استفاده از یک کد که به زبان پایتون[۱۹] نوشته شد, اقدام به تحلیل داده‌های ذخیره شده کردیم. ۴ معیار اصلی که اندازه‌گیری شد شامل موارد زیر می‌باشند:

* درصد منطبق[۲۰] بودن دو صفحه بازی : این معیار را به این صورت تعریف کردیم که درصد مواقعی که رخدادهای در حال انجام در دو طرف بازی به یک‌شکل هستند. برای مشخص‌تر شدن این مورد یک مثال می‌زنیم. فرض کنید که یک برخورد در سمت بازیکن ۱ صورت گرفته است, طبیعتا می‌دانیم که در لحظه برخورد صرفا همان بازیکن اطلاعات بروز را دارا است اما در صورتی که بازیکن شماره ۲ نیز اطلاعات برخورد را دریافت کرده و در حال اعمال کردن تغییرات مورد نظر باشد در این حالت دو سمت بازی منطبق هستند. در صورتی که این اطلاعات برخورد به هر دلیلی به سمت مقابل نرسیده و تغییرات لازم در حال اعمال شدن نباشند آنگاه در نظر می‌گیریم که دو طرف به طور موقتی از حالت انطباق خارج شده‌اند.

* نرخ دریافت بسته‌ها از سمت دو بازیکن : همان‌طور که از نام این معیار مشخص است, می‌توان با استفاده از آن درصد بسته‌هایی که به درستی از تونل ارتباطی ما به طرف دیگر می‌رسند را محاسبه کرد. در میان منابع

[19] Python
[20] Sync

موجود در اینترنت توضیح دقیقی درباره قابل اعتماد[21] بودن راه ارتباطی بلوتوث در اندروید داده نشده است لذا محاسبه این معیار دارای اهمیت است. البته لازم به ذکر است که نوع پیاده‌سازی OBEX که توضیحات آن در Google Play در دسترس است وجود دارد که نوعی پروتکل ارتباطی بوده و تضمین بیشتری برای فرایند ارسال و دریافت می‌دهد اما در این پروژه پیاده‌سازی این مورد سربازی برای سیستم داشت که ارزش کارکردی قابل توجهی نیز به تجربه کاربری نهایی اضافه نمی‌کرد.

* میانگین زمان ارسال داده‌ها : برای محاسبه این مورد هرگاه که پیامی ارسال می‌شود یک تایمر نیز آغاز شده و هنگامی که تاییدیه این پیام از سمت دیگر دریافت شده زمان کل نصف شده تا زمان ارسال داده محاسبه شود. سپس تمام این زمان‌ها با در کنارهم در نظر گرفته شده تا میانگین زمان ارسال معلوم شود.

* بیشینه تاخیری که بین ارسال و دریافت داده‌ها مشاهده شد

- **نحوه اجرای تست (پیاده‌سازی)**

برای تست کردن پروژه همان‌طور که تا کنون مطرح شد اطلاعات ارتباط بین دو دستگاه را هنگام بازی ذخیره کرده و سپس تحلیل کردیم. شکل زیر بخشی از اطلاعات مربوط به فایل log می‌باشد تا با کلیات ساختار نگه‌داری داده‌ها آشنا شویم:

```
receiveBallCollisionMessage : 200 0.780556 0.922639 0.647222 0.453251 1626.0
sendBallCollisionAck : 200 R858.0@
receiveBallCollisionMessage : 201 0.780556 0.922639 0.648148 0.452776 1628.0
sendBallCollisionAck : 201 R859.0@
receiveBallCollisionMessage : 201 0.780556 0.922639 0.648148 0.452776 1628.0
sendBallCollisionAck : 201 R860.0@
receiveBallCollisionMessage : 202 0.780556 0.922639 0.648148 0.452776 1629.0
sendBallCollisionAck : 202 R861.0@
sendBallCollision : 286 C0.639815#0.497213#0.775000#0.311037#862.000000@
receiveCollisionAckMessage : 288 1630.0
sendStrikerPosition : 326 P0.530556#0.605909#863.000000@
sendStrikerPosition : 327 P0.529630#0.630992#864.000000@
```

شکل ۲۰ – ساختار فایل‌های Log پروژه

[21] Reliable

حال به توضیح نحوه تحلیل این فایل‌ها در کدی که به زبان پایتون زده شده است می‌پردازیم. در شکل زیر تابع مربوط به ساختن ساختار داده‌های لازم برای اطلاعات مختلفی که ذخیره شده‌است را در کنار توابع لازم برای بررسی کمی اطلاعات مشاهده می‌کنیم. برای نمونه, بدنه یکی از توابع نیز نشان‌داده شده است:

```
> def createLogDataStructure(): ···

> def createStrikerPositionLog(frame,x,y,id): ···

> def createBallCollisionLog(frame,x,y,vx,vy,id): ···

v def createBallCollisionAckLog(frame,id):
v     return {
          "type" : "BallCollisionAck",
          "msgId" : float(id),
          "frame" : float(frame)
      }

> def createGoalScoredAckLog(frame,goal,id): ···

> def createGoalScoredLog(frame,goal,id): ···
```

شکل ۲۱ – توابع مقدماتی تحلیل تست

سپس سه تابع دیگر تعریف شده‌اند که به عنوان Utility کاربرد دارند و هدف هر کدام نیز به سادگی از روی نامشان قابل تشخیص است:

```
> def is_digit(str): ···

> def parseLine(line : str): ···

> def readFile(file): ···
```

شکل ۲۲ – توابع کاربردی تحلیل تست

دو تابع اصلی نیز برای انجام محاسباتی که نتایج آنان در بخش بعدی قابل مشاهده است استفاده شد. این توابع به نام‌های checkBallSync و checkDelay هستند که وظیفه هر کدام نیز از روی نامشان مشخص است. پیاده‌سازی انجام شده برای هریک از این توابع در فایل موجود در ضمیمه این گزارش قابل مشاهده بوده و صرفا توضیح اجمالی از نحوه کار کردن هر کدام در این بخش ارائه می‌دهیم.

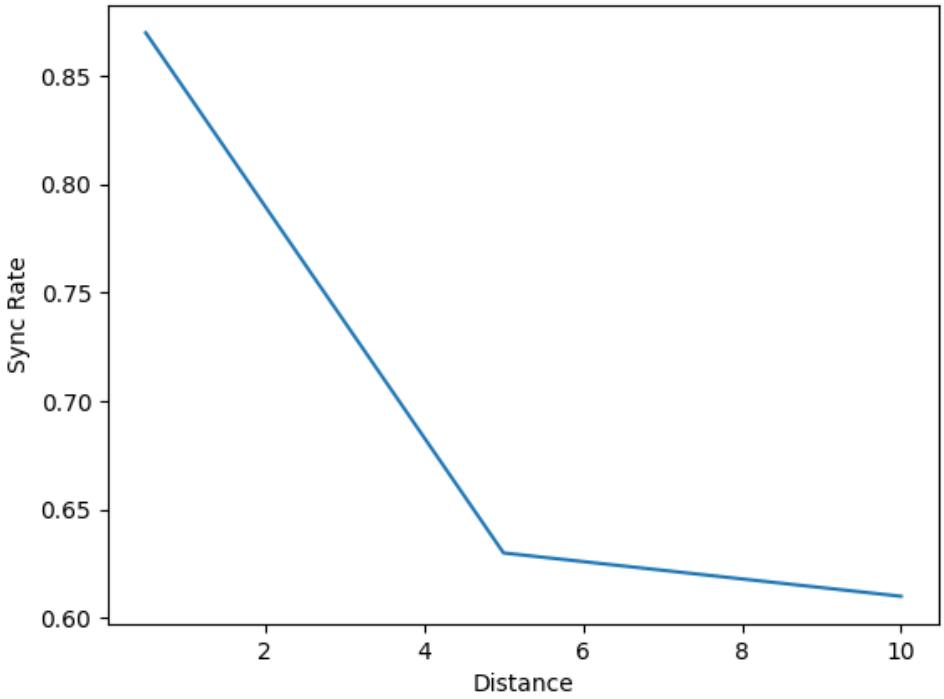
- تابع checkBallSync : این تابع برای اندازه‌گیری درصد انطباق دو دستگاه در حین ارتباط است. بدین صورت کار می‌کند که با استفاده از ساختارداده Set تعداد فریم‌هایی که دو پیام در دو سمت با یکدیگر منطبق نبوده‌اند را در نظر گرفته و این مورد را تقسیم بر کل تعداد فریم‌هایی که در طول بازه موجود برای آن دو پیام تعریف شده است کرده‌ایم تا درصد منطبق نبودن را حساب کرده و سپس از روی آن بتوانیم درصد انطباق را برای آن دو پیام و در نهایت برای کل فرایند حساب کنیم.
- تابع checkDelay :این تابع برای اندازه‌گیری تاخیر بین ارسال و دریافت داده استفاده می‌شود. همان‌طور که در مفاهیم پایه‌ای شبکه‌های کامپیوتری نیز برای محاسبه تاخیر از فرایند ارسال داده و دریافت تاییدیه ارسال بهره‌برداری می‌شود, در این تابع نیز یک تایمر زمان بین ارسال داده به طرف دیگر و دریافت کردن تاییدیه مشاهده داده ارسالی را در نظر گرفته و این زمان سپری شده را تقسیم بر ۲ می‌کند تا صرفا زمان ارسال محاسبه شود. (با این فرض که زمان ارسال و دریافت به تا حد خوبی باهم برابر می‌باشند)

نتایج تست‌های انجام شده

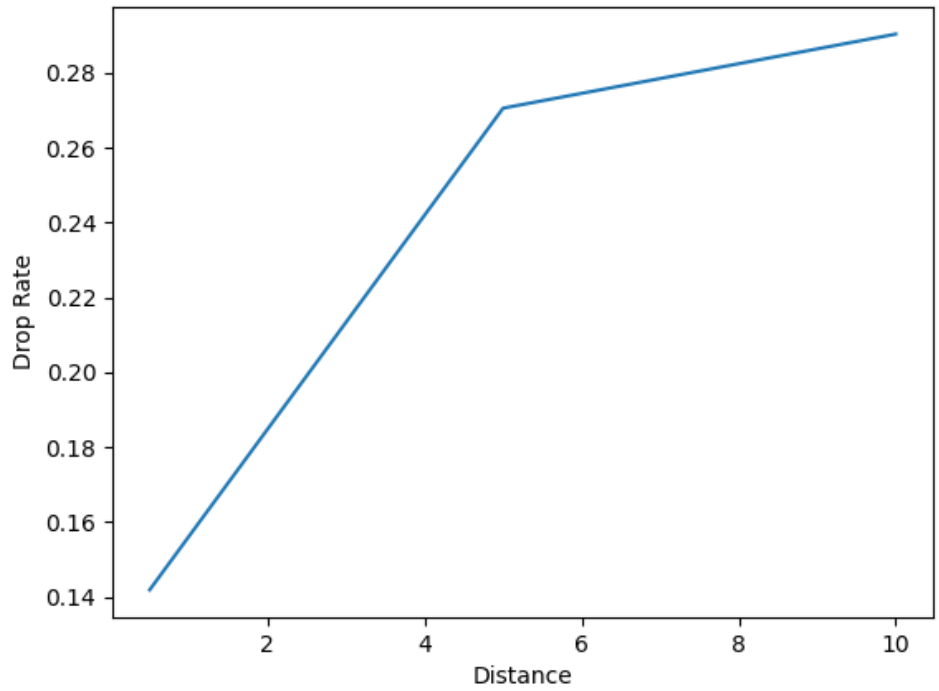
مواردی که در قسمت‌های قبلی توضیح داده شده‌اند در سه حالت اندازه‌گیری شدند. در فاصله معمول و نزدیک که ۵۰ سانتی‌متر بین دو دستگاه فاصله باشد, حالت مشابه ولی فاصله ۵ متری و در نهایت نیز بازی در حالتی که دو طرف ۱۰ متر از یکدیگر فاصله دارند و این ۱۰ متر نیز نهایت برد توصیه شده برای ارتباط بلوتوث بین دو دستگاه اندروید می‌باشد. نتایج را ابتدا در جدول زیر مشاهده کرده و سپس نمودارهایی را برای درک بهتر آنان در ادامه خواهیم دید:

	درصد انطباق	نرخ دریافت داده	میانگین تاخیر(ms)	بیش‌ترین تاخیر
۰.۵ متر	۰.۸۷۸۱	۰.۸۵۹۲	۳۰.۹	۶۸
۵ متر	۰.۶۳۷	۰.۷۲۹	۳۴.۶۵	۶۸
۱۰ متر	۰.۶۱۳	۰.۷۰۹	۳۸.۷۲	۷۶.۵

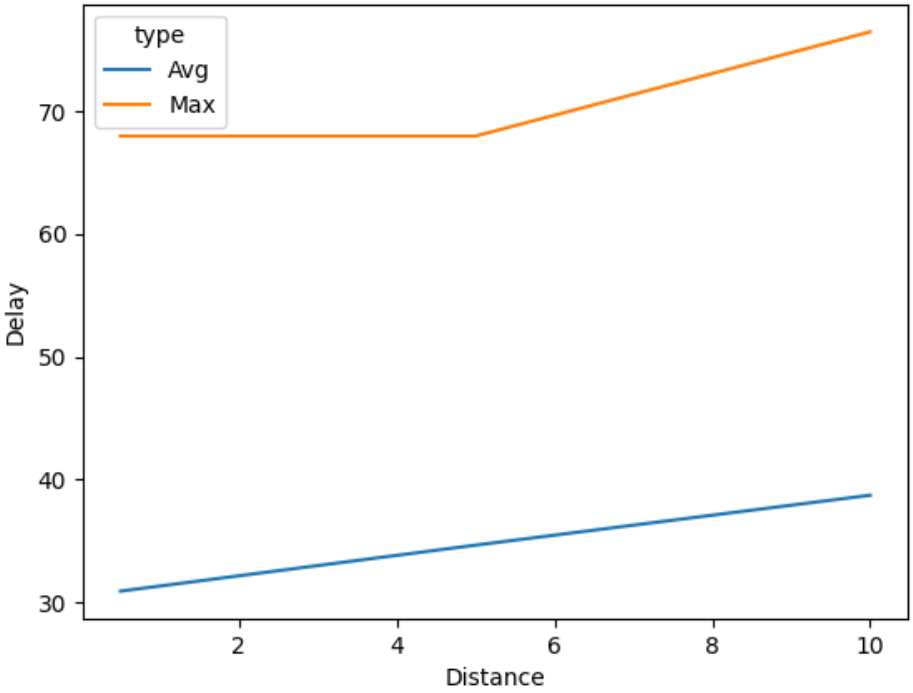
شکل‌های تولید شده از اعداد موجود در جدول بالا نیز به شرح زیر هستند:



شکل ۲۳ – درصد انطباق برای فواصل مختلف



شکل ۲۴– میانگین نرخ از دست رفتن داده برای فواصل مختلف



شکل ۲۵ –زمان تاخیر ارسال و دریافت برای فواصل مختلف

- **تحلیل نتایج**

همان‌طور که انتظار می‌رفت, با افزایش فاصله از حدود ۵۰ سانتی‌متر که به نوعی کم‌ترین فاصله معمول یک بازی موبایلی دونفره تا حدود ۱۰ متر که بیش‌ترین فاصله‌ای است که ارتباط بلوتوث از لحاظ تئوری در آن کار می‌کند, به طور معمول شاهد رخدادهای زیر می‌باشم:

* درصد انطباق بین دو دستگاه کمتر می‌شود.
* نرخ از دست رفتن بسته‌های ارسال شده از یک طرف به طرف دیگر بالاتر می‌رود.
* میانگین زمان تاخیر مشاهده شده بین ارسال و دریافت داده‌ها بالاتر می‌رود.

به طور منطقی نیز می‌توان نتیجه‌گیری کرد که کیفیت ارتباط بلوتوث (که با معیارهای مختلف اندازه‌گیری و گزارش شد) رابطه‌ای مستقیم با میزان فاصله دو دستگاه استفاده‌کننده از آن دارد.

۸- پاسخ به سوالات طراحی و تایید شده در پروپوزال

در ابتدا برای یادآوری سوالاتی که در فرم نهایی پیشنهاد پروژه مطرح و تایید شدند را در این بخش شرح می‌دهیم:

- بررسی اینکه مکان توپ در هر دو دستگاه با یکدیگر همگام است یا خیر؟ (اول)
- تاثیر فاصله مکانی دو دستگاه در مقدار تاخیر ایجاد شده در بازی چه مقدار است؟ (دوم)
- میزان تاخیری که قسمت مربوط به منطق بازی ایجاد می‌کند چقدر است؟ (سوم)

همان‌طور که تا کنون توضیح داده شد, مورد اول همان میزان منطبق بودن دو دستگاه با یکدیگر است که به طور کامل در بخش تست پروژه تحلیل شده و مقادیر مربوط به آن گزارش شده‌اند. مورد دوم نیز به همین صورت بوده و توضیحات آن در کنار نمودارهای معمول در بخش تست در دسترس است. برای بخش سوم از مقادیر محاسبه شده برای تاخیر که در بخش تست موجود است استفاده کرده و با گذاشتن متغیرهای اضافه در سطح Log به این نتیجه رسیدیم که تاخیر مربوط به انجام محاسبات بازی به طور میانگین در حدود ۱ میلی‌ثانیه می‌باشد.

۹- پیوست‌های فنی

در این بخش از پیوست‌هایی که برای بخش‌های مختلف طراحی و پیاده‌سازی این پروژه انجام شد نام برده شده است. این پیوست‌ها در قالب‌های مختلفی مانند مقالات موجود در بلاگ‌ها, ویدیو و مستندات[۲۲] می‌باشند:

Video contents:

- Youtube#1
- Youtube#2
- Youtube#3
- Youtube#4

[22] Documentation

Blogs and guides:

- Medium#1
- Medium#2

Android Documentations:

- Android#1
- Android#2

١٠- مراجع

[1] Wang, E. (2010). Experiences from Implementing a Mobile Multiplayer Real-Time Game for Wireless Networks with High Latency. *International Journal of Computer Games Technology, 2009, 530367.*

[2] Zhang, Y., Martikainen, O., Pulli, P., & Naumov, V. (2011). Real-Time Process Data Acquisition with Bluetooth. In *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*. Association for Computing Machinery.

[3] Syafrudin, M., Lee, K., Alfian, G., Lee, J., & Rhee, J. (2018). Application of Bluetooth Low Energy-Based Real-Time Location System for Indoor Environments. In *Proceedings of the 2018 2nd International Conference on Big Data and Internet of Things* (pp. 167–171). Association for Computing Machinery.

[4] DiMarzio, Jerome F. Android Studio Game Development. Apress, 2015.

[5] Pruett, Chris. "Writing real time games for Android." Vortag Google IO 6 (2009).