

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ



دانشگاه صنعتی جندی‌شاپور دزفول  
دانشکده برق و کامپیوتر  
گروه کامپیوتر

پایاننامه برای دریافت درجه کارشناسی  
در رشته کامپیوتر گرایش سخت افزار

## ساخت بازی واقعیت مجازی

استاد راهنما:

جناب دکتر شکیبا

استاد مشاور:

جناب دکتر شکیبا

نگارش:

سمانه شیرین نژاد

پاییز 1395

## تأییدیه‌ی صحت و اصالت نتایج

### بسمه تعالی

اینجانب سمانه شیرین نژاد به شماره دانشجویی \*\*\*\*\*\* دانشجوی رشته کامپیوتر مقطع تحصیلی کارشناسی تأیید می‌نمایم که کلیه‌ی نتایج این پایاننامه/رساله حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر نموده‌ام. درصورت اثبات خلاف مندرجات فوق به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انسباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقة حقوق مکتب و تشخیص و تعیین تخلف و مجازات را از خوبیش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذیصلاح (اعم از اداری و قضایی) به عهده‌ی اینجانب خواهد بود و دانشگاه هیچ‌گونه مسؤولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی:

امضا و تاریخ:

## تقدیم به:

پدر و مادر عزیز و مهربانم که در سختی‌ها و دشواری‌های زندگی همواره یاوری دلسوز و فداکار و پشتیبانی محکم و مطمئن برایم بوده‌اند.

**تقدیر:**

از استاد گرامیم جناب آقای دکتر شکیبا بسیار سپاسگزارم چرا که بدون راهنمایی های ایشان تأمین این پایان نامه بسیار مشکل می نمود.

## فهرست مطالب

	عنوان	صفحه
2	<b>فصل 1: مقدمه</b>	
7	<b>فصل 2: ساخت بازی واقعیت مجازی در یونیتی</b>	
8	<b>مقدمه</b>	
8	2-1- پلتفرم واقعیت مجازی گوگل چگونه عمل می کند .....	
8	2-2- کیت توسعه نرم افزار ویژه واقعیت مجازی .....	
9	2-3- ساخت بازی در یونیتی .....	
10	2-3-2- ایجاد پروژه جدید و انجام تنظیمات اولیه محیط بازی در یونیتی .....	
14	2-3-3- تنظیمات دوربین .....	
17	2-3-4- آجکت‌های تعاملی در بازی .....	
19	2-3-5- اسکریپت تعامل .....	
26	2-3-6- چگونه تعامل اتفاق می‌افتد .....	
27	2-3-7- خروجی گرفتن و اجرای بازی بر روی هدست واقعیت مجازی .....	
29	<b>فصل 3: ارتباط دستگاه‌های اندروید از طریق WiFiP2P</b>	
30	3-1- مقدمه .....	
30	3-2- API های وای-فای .....	
30	3-3- WifiP2pManager کلاس .....	
31	3-4- ایجاد یک دریافت کننده انتشاری برای Intent های وای-فای P2P .....	
32	3-5- ایجاد یک اپلیکیشن وای-فای P2P .....	
33	3-5-1- تنظیمات اولیه .....	
35	3-5-2- شناسایی جفت‌ها .....	
37	3-5-3- اتصال به جفت‌ها .....	
37	3-5-4- انتقال داده .....	
44	<b>فصل 4: نتیجه‌گیری و پیشنهادها</b>	
45	4-1- مقدمه .....	
45	4-2- محتوا .....	
45	4-2-1- جمع‌بندی .....	
46	4-2-2- نوآوری .....	
46	4-2-3- پیشنهادها .....	
47	<b>فهرست منابع</b>	

## فهرست جداول

	عنوان	صفحة
31 .....	جدول (1-3) متدهای وای-فای P2P	جدول
31 .....	جدول (2-3) شنوندهای وای-فای P2P	جدول
31 .....	جدول (3-3) های وای-فای Intent P2P	جدول

## فهرست اشکال

عنوان	
صفحه	
شكل(1-1) یک نمونه عینک واقعیت مجازی .....	3
شكل(1-2) عینک واقعیت مجازی گوگل .....	5
شكل (3-1) عینک واقعیت مجازی را خودتان بسازید! .....	6
شكل (1-2) نمایی از موتور بازی ساز یونیتی .....	10
شكل (2-2) مشخص کردن مسیر نصب کیت توسعه نرم افزار اندروید در یونیتی .....	11
شكل (3-2) تصویری از بازی در حالت اجرا هنگامی که از GvrViewerMain استفاده می شود ..	14
شكل (4-2) متغیرهای مورد نیاز در اسکریپت راه رفتن خودکار .....	16
شكل (5-2) تابع شروع در اسکریپت راه رفتن .....	16
شكل (6-2) تابع بروزرسانی در اسکریپت راه رفتن خودکار .....	17
شكل(7-2) تنظیمات اسکریپت راه رفتن خودکار در پنجره Inspector .....	17
شكل (8-2) مؤلفه Collider به صورت خطوط سبز رنگ اطراف آبجکت مشخص می شود .....	18
شكل (9-2) تنظیمات مؤلفه Collider برای یک آبجکت .....	19
شكل(10-2) تنظیمات مؤلفه Animator برای کاراکتر Crawler .....	20
شكل (11-2) ایجاد ماشین حالت در Animator یونیتی .....	21
شكل (12-2) متغیرهای مورد نیاز در اسکریپت تعامل با Crawler .....	21
شكل (13-2) تابع پاسخ به تعامل با کاراکتر Crawler .....	22
شكل (14-2) بخش های مختلف تنظیمات سیستم ذره .....	23
شكل (15-2) ایجاد افکت آتش با استفاده از سیستم ذره یونیتی .....	25
شكل (16-2) اسکریپت تعامل برای آبجکت Cauldron .....	25
شكل (17-2) اضافه کردن مؤلفه Event Trigger به آبجکت تعاملی .....	26
شكل(18-2) مؤلفه های اضافه شده به آبجکت Event System .....	27
شكل (19-2) اجرای بازی در نمایشگر واقعیت مجازی گوگل .....	28
شكل (1-3) ایجاد یک دریافت کننده انتشاری .....	32
شكل (2-3) مرحله اول از تنظیمات اولیه اپلیکشن وای-فای P2P .....	33
شكل (3-3) مرحله دوم از تنظیمات اولیه اپلیکشن وای-فای P2P .....	33
شكل (4-3) مرحله سوم از تنظیمات اولیه اپلیکشن وای-فای P2P .....	34
شكل (5-3) مرحله چهارم از تنظیمات اولیه اپلیکشن وای-فای P2P .....	34
شكل (6-3) مرحله پنجم از تنظیمات اولیه اپلیکشن وای-فای P2P .....	35
شكل(3-7) فراخوانی تابع discoverPeers() .....	36
شكل (8-3) تنظیم WIFI_P2P_PEERS_CHANGED_ACTION .....	36
شكل (9-3) اطلاع از موفقیت یا عدم موفقیت در برقراری اتصال .....	37
شكل (10-3) انتقال داده بین کلاینت و سرور .....	39
شكل (11-3) انتقال فایل JPEG .....	40

## چکیده

مفهوم واقعیت مجازی از دو کلمه‌ی «واقعیت» و «مجازی» می‌آید. واژه مجازی را می‌توان به صورت «مانند» یا «نزدیک به» معنی کرد و معنی واژه واقعیت در حقیقت همان واقعیتی است که بشر تجربه می‌کند. بنابراین اصطلاح واقعیت مجازی اساساً به معنی «چیزی شبیه به واقعیت» است. بنابراین واقعیت مجازی مستلزم نمایش صحنه‌های مجاز با استفاده از کامپیوتری است که محیط مجازی را که می‌توانیم در آذبه کاوش بپردازیم تولید می‌کند. در حقیقت عبارت واقعیت مجازی برای توصیف محیط سه بعدی تولید شده توسط کامپیوتر استفاده می‌شود. شخصی که به عنوان عضوی از این دنیای مجازی است درون یک محیط مجازی غوطه‌ور شده و تا زمانی که آنجاست قادر است تا آبجکت‌هایی را دستکاری کرده و یک مجموعه از اعمال را اجرا کند. در این مقاله قصد داریم تا یک بازی واقعیت مجازی با استفاده از موتور بازی ساز یونیتی برای گوشی‌های هوشمند اندروید بسازیم و با استفاده از عینک واقعیت مجازی گوگل، محیط مجازی ساخته شده را مشاهده نموده، در آذبه کاوش بپردازیم و با یکسری آبجکت‌های خاص تعامل داشته باشیم.

**واژه‌های کلیدی:** واقعیت مجازی، VR، موتور بازی ساز یونیتی، عینک واقعیت مجازی گوگل، مقوای گوگل.

**فصل اول**

**مقدمه**

## مقدمه

واقعیت مجازی<sup>۱</sup> که به اختصار با حروف VR نمایش داده می‌شود یک تکنولوژی و فناوری نوین است

که به کاربر امکان می‌دهد تا با یک محیط شبیه‌سازی رایانه‌ای اندرکنش یا تعامل داشته باشد. در واقع در آن محیطی مجازی در جلوی چشم‌انداز کاربر قرار می‌گیرد و براساس حرکت سر و بدن با آن محیط مجازی تعامل برقرار می‌کند. به عبارت دیگر هنگامی که یک فرد هدست واقعیت مجازی را بر روی سر خود نصب می‌کند، در جلوی چشم‌انداز خود محیطی را مشاهده می‌کند که براساس تغییر موقعیت بدنش تغییر می‌کند و ذهن انسان پس از مدتی می‌پذیرد که در یک محیط واقعی قرار گرفته است.

یک محیط واقعیت مجازی در هدست واقعیت مجازی توسط اپلیکیشن‌های اختصاصی آن به وجود می‌آید. برخی از این محیط‌ها به صورت گرافیک رایانه‌ای و سه بعدی هستند و برخی دیگر نیز ویدئوها یا تصاویری 360 درجه از محیط‌های واقعی هستند که از قبل فیلم‌برداری شده‌اند. با این قابلیت فناوری واقعیت مجازی می‌توان این امکان را فراهم کرد تا افراد بتوانند از امکانات و مکان‌شما به خوبی دیدن کنند. اغلب محیط‌های واقعیت مجازی در درجه اول تجربه‌های دیداری می‌باشند که از طریق یک هدست واقعیت مجازی قابل مشاهده و تجربه می‌باشند. برخی از اپلیکیشن‌ها دارای اطلاعات حسی دیگری مانند تولید صدا هم می‌باشند. محیط‌های شبیه‌سازی شده می‌توانند مانند محیط‌های زندگی واقعی و یا به صورت کاملاً متفاوت باشند نظیر آنچه در محیط‌های بازی دیده می‌شود.



شکل (1-1) یک نمونه عینک واقعیت مجازی

یکی از کاربردهای مهم فناوری واقعیت مجازی در گردشگری است. به این صورت که ویدئوها و یا تصاویری 360 درجه و با کیفیت بالا از مکان‌های توریستی و مهم تهییه می‌شود و افراد هزاراد کیلومتر آنطرف‌تر می‌توانند با استفاده از هدست‌های واقعیت مجازی آن ویدئوها را آنگونه ببینند که گویی در همانجا قرار دارند و به هر طرف که بخواهند می‌توانند حرکت کنند. یکی دیگر از کاربردهای این فناوری نمایش همه

<sup>1</sup> Virtual Reality

جهته و کامل فضای داخلی و بیرونی مکانهایی است که برای فروش یا اجاره قرار گرفته‌اند. به طوری که فرد بیننده به راحتی می‌تواند بدوز حضور در آن مکان همه جنبه‌های آن را از طریق هدست واقعیت مجازی بررسی کرده و انتخاب کند. همچنین از این فناوری می‌توان برای پیش نمایش طرح‌های نمای داخلی آپارتمانها استفاده نمود. اگر صنعت بازی سازی در حال حاضر یکی از کاربردهای اولیه و اصلی تکنولوژی واقعیت مجازی نباشد، پس دیگر شکل‌های سرگرمی قطعاً خواهد بود. در حال حاضر تماشاگران سینما می‌توانند از فیلم‌های سه‌بعدی لذت ببرند؛ اما با اپلیکیشن‌های ویژه‌ای کاربراد می‌توانند به شکلی عمیق‌تر و موثرتر از دیده‌این فیلم‌ها لذت ببرند. آنها می‌توانند فیلم‌ها را روی یک پرده بسیار بزرگ مجازی تماشا کنند – چیزی شبیه به یک تجربه شخصی دیده‌فیلم- و خودشان را درون آن فیلم تصور کنند؛ محاصره شده در میان افکت‌های تصویری و صوتی. حتی در صنعت توریسم و جهانگردی هم این تکنولوژی نفوذ کرده است. از دیگر کاربردهای آن می‌توان به استفاده در صنعت بهداشت و درمان اشاره نمود. صنعت بهداشت و درمان با وجود مؤسسه‌اتی که از تصاویر کامپیوترا برای تشخیص و درمان بیماری‌ها استفاده می‌کنند، یکی از بزرگ‌ترین استقبال‌کنندگان از تکنولوژی واقعیت مجازی است. شبیه‌سازی‌های واقعیت مجازی می‌توانند تصاویر تشخیصی درست و دقیقی را از اسکن‌ها و مدل‌های سه‌بعدی تهیه شده از آناتومی بیماران را ایجاد کنند. مدل‌های مجازی به پزشکان و جراحان تازه‌کار و با تجربه کمک می‌کنند تا بتوانند امن‌ترین و موثرترین راه برای پیدا کردن محل تومورها را ارائه دهند، ابزارهای جراحی را در جای درست قرار دهند یا امکان تمرین و تست درمانها و عمل‌های جراحی پیچیده را فراهم بیاورند. صرف‌نظر از جراحی، واقعیت مجازی همچنین می‌تواند به عنوان یک ابزار موثر و مفروز به صرفه برای توانبخشی به کار بrede شود. بیماران ضایعات مغزی و نخاعی در سراسر اروپا حالاً می‌توانند از یک درمان مؤثر مبتنی بر واقعیت مجازی بهره ببرند. تمرینات مجازی و بازخوردهای در لحظه احساس انجام یک بازی را در بیمار ایجاد کرده و با ایجاد انگیزه در بیماران آنها را به تمرین هر روزه ترغیب می‌کند.

واقعیت مجازی می‌تواند علايق فرهنگی ما را نیز پررنگ‌تر کند. این تکنولوژی می‌تواند بلافضله و در لحظه کاربران را به موزه لور پاریس، آکروبولیس در آتن و موزه گوگنهایم در نیویورک ببرد. در واقع تعدادی از موزه‌ها در حال حاضر با همکاری توسعه‌دهندگان توanstه‌اند فضاهای مجازی را ایجاد کنند که مردم بتوانند تجربه حضور فیزیکی در موزه را در آنها داشته باشند. سال‌گذشته موزه بریتیش در لندن اولین آخر هفت‌های واقعیت مجازی خودش را برگزار کرد و موزه تاریخ طبیعی آمریکا در شهر نیویورک هم امکان تماشای بعضی از مجموعه‌های خودش را به صورت مجازی و از طریق عینک واقعیت مجازی گوگل برای علاقه‌مندان فراهم آورد. به این ترتیب حالاً هر کاربر موبایل هوشمند می‌تواند با استفاده از عینک واقعیت مجازی گوگل به تماشای این موزه برسد. و در آخر می‌توان گفت این تکنولوژی در فضانوری نیز کاربرد دارد. این تکنولوژی در آینده‌ای نه چندان دور می‌تواند برای کنترل‌های نوردها یا دیگر ابزارها از میلیون‌ها مایل دورتر به کار بrede شود.

بر اساس نوع کارکرد هدست‌های واقعیت مجازی می‌توان آنها را به دو بخش تقسیم نمود. اول هدست‌هایی که برای نمایش تصویر از نمایشگرهای اختصاصی درون خود استفاده می‌کنند و دوم هدست‌هایی که برای نمایش تصویر از نمایشگر تلفن همراه هوشمند استفاده می‌کنند. با توجه به اینکه تلفن‌های همراه پرقدرت که قابلیت پردازش و نمایش اپلیکیشن‌های واقعیت افزوده را داشته باشند در حال حاضر زیاد شده‌اند لذا هدست‌هایی که بر این اساس کار می‌کنند نیز بسیار بیشتر مورد استقبال قرار گرفته‌اند. در حال حاضر اکثر هدست‌های واقعیت مجازی در بازارهای جهانی مبتنی بر تلفن‌های هوشمند هستند و به همین خاطر هم

قیمت بسیار پایین‌تری نسبت به نوع اول هدست‌ها دارد. در این پروژه ما از عینک واقعیت مجازی گوگل با نام مقوای گوگل<sup>1</sup> که از هدست‌های نوع دوم است استفاده می‌کنیم.

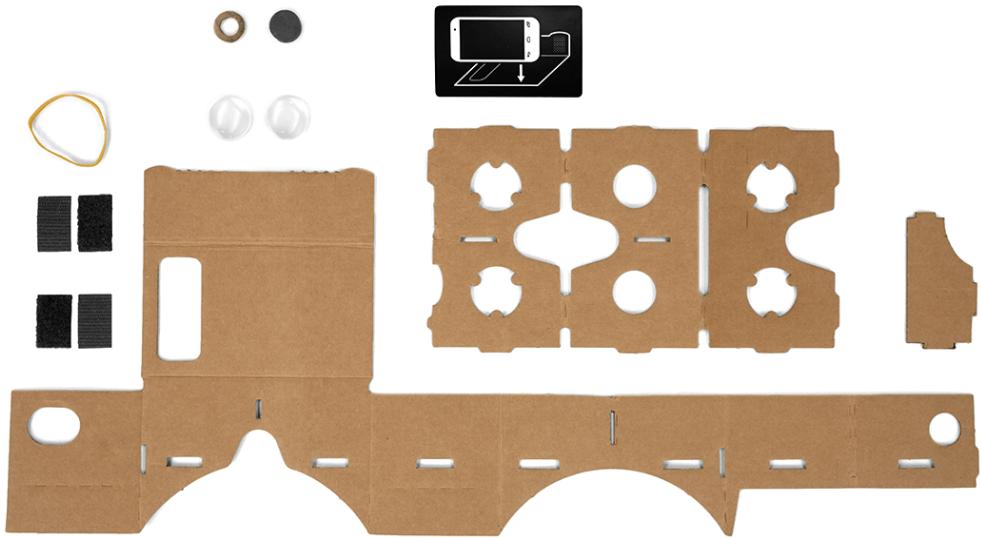


شکل (1-2) عینک واقعیت مجازی گوگل

این نمونه هدست یکی از ارزان‌ترین نمونه‌های عینک واقعیت مجازی موجود در بازار است زیرا برای ساخت آن همانطور که از نامش پیداست از مقوای استفاده شده است و از طرفی هم از گوشی هوشمند برای اجرای اپلیکشن واقعیت مجازی بهره می‌برد. در حقیقت ایده دسترسی تعداد زیادی از مردم به واقعیت مجازی شرکت گوگل را به سمت استفاده از گوشی‌های هوشمند برای این هدف جذب کرد. از دیگر مزایای آن می‌توان به سبک بودنش اشاره کرد که برای کارهای آزمایشگاهی و تست اپلیکشن بسیار خوش‌دست است. گوگل در کنفرانس I/O در سال 2014 از هدست واقعیت مجازی مقوایی خود پرده‌برداری کرد. در این هدست از دو عدد لنز یا عدسی دو طرف محدب استفاده شده است که این امکان را برای کاربر فراهم می‌کند تا با قرار دادن گوشی هوشمند در آن و اجرای اپلیکشن واقعیت مجازی بتواند محیط را درست مانند واقعیت شبیه‌سازی شده ببیند. همچنین هر کاربری می‌تواند با تهیه این عدسی‌ها و با داشتن مقوای، آهنربا، نوار چسب و کش نمونه مورد نظر خودش را بسازد!

---

<sup>1</sup> Google Cardboard



**شکل (1-3) عینک واقعیت مجازی را خود تازه بسازیدا**

در این پروژه قصد داریم تا یک بازی واقعیت مجازی با استفاده از موتور بازی ساز یونیتی<sup>1</sup> برای سیستم عامل اندروید بسازیم و آنرا از طریق اسمارت‌فون اندروید خود و عینک واقعیت مجازی گوگل مشاهده کنیم. همچنین امکان ساخت برای سیستم عامل iOS نیز وجود دارد.

---

<sup>1</sup> Unity

## **فصل دوم**

### **ساخت بازی واقعیت مجازی در یونیتی**

## مقدمه

مقوای گوگل تکنولوژی است که اپلیکیشن‌های واقعیت مجازی را بر روی گوشی‌های هوشمند توسعه می‌دهد. عنصر اصلی این پلتفرم یک هدست مقوای ارزان قیمت به همراه دو عدد لنز است که یک «نمای سه بعدی برجسته بینی<sup>۱</sup>» از صحنه‌ی نمایش داده شده بر روی صفحه نمایش گوشی را ارائه می‌دهد. هدف از این فصل ارائه نحوه ساخت یک بازی واقعیت مجازی سازگار با مقوای گوگل و سیستم عامل اندروید توسط موتور بازی ساز یونیتی است.

### 2-1- پلتفرم واقعیت مجازی گوگل چگونه عمل می‌کند

ایجاد یک تجربه واقعیت مجازی در تئوری، بسیار آسان است. به جای نمایش جهاز به عنوان یک تصویر مفرد، دو تصویر نمایش می‌دهیم. این تصاویر از دوربین‌هایی می‌آیند که به فاصله‌ی چند سانتی متر از هم قرار گرفته‌اند و کاربر تصویر را با چشم چپ از دوربین سمت چپ و با چشم راست از دوربین سمت راست می‌بیند و در نتیجه‌ی آن نمودی از عمق ایجاد می‌شود. افزون بر آن با استفاده‌ی درست از سنسورهای حرکتی می‌توان جهتی که کاربر با آن مواجه است شناسایی و حرکت سر کاربر را نیز مسیریابی کرد. در نهایت ترکیب آن با دنیای سه بعدی که ساخته‌اید یک تجربه واقعیت مجازی بی‌نظیر را فراهم می‌آورد. در عمل، این امر به سخت افزار نسبتاً پیچیده‌ای نیاز دارد تا بتواند در کنار مسیریابی حرکت سر کاربر، دو تصویر بر روی یک صفحه باوضوح بالا نمایش داد و همه‌ی این موارد را در یک دستگاه کوچک و به اندازه کافی سبک قرار داد تا سبب آزار گردن تاز نشود!

ایده‌ای که پشت پلتفرم واقعیت مجازی گوگل با نام مقوای گوگل قرار دارد، استفاده از گوشی‌های هوشمند برای این هدف است، بدین ترتیب که با استفاده از سنسورها و نمایشگر گوشی به همراه مقداری مقوا و یک جفت لنز آن را به یک دستگاه واقعیت مجازی تبدیل می‌کند.

### 2-2- کیت توسعه نرم افزار ویژه واقعیت مجازی

گوگل تا کنون دو پلتفرم واقعیت مجازی Cardboard و Daydream را عرضه کرده و برای پلتفرم نوع دوم عینک مقوای خود را ارائه نموده است که در این پروژه مورد بحث ما است. گوگل خصوصیاتی سازگار با هدست‌ها منتشر کرده و بدین منظور دو کیت توسعه نرم افزار<sup>۲</sup> یا SDK یکی برای سیستم عامل اندروید که قرار است بر روی آن بازی واقعیت مجازی اجرا شود و دیگری برای موتور بازی ساز یونیتی که می‌خواهیم بازی واقعیت مجازی را بوسیله آن ایجاد کنیم عرضه نموده است. بنابراین در این پروژه به این دو کیت توسعه نرم افزار نیاز داریم و پیش از ایجاد یک پروژه بازی در یونیتی لازم است تا از قبل آنها را دانلود کرده باشیم.

<sup>1</sup> Stereoscopic 3D View

<sup>2</sup> Software Development Kit

کیت توسعه نرم افزار واقعیت مجازی گوگل حاوی کتابخانه‌ها، استناد API، رهنمودهای طراحی و نمونه‌هایی است که به توسعه‌دهندگان جهت ساخت بازی واقعیت مجازی کمک می‌کند. به عبارت دیگر به ما امکان ساخت اپلیکیشن‌هایی را می‌دهد که بتوان صحنه‌های سه بعدی را با استفاده از رندرینگ دو چشمی<sup>۱</sup> نمایش دهیم، صدای سه بعدی رندر کنیم به این صورت که میزان صدا وابسته به فاصله‌مان از منبع صدا است و همچنین حرکات سر را ردیابی کرده و نسبت به آذون‌العمل لازم را نشان می‌دهد. در زیر به چند نمونه از امکاناتی که این کیت برای ما فراهم می‌آورد اشاره می‌کنیم:

- مسیریابی حرکت سر کاربر
- رندرینگ استریو پهلو به پهلو
- شناسایی تعامل کاربر با سیستم
- تصحیح انحراف لنزهای نمایشگر واقعیت مجازی
- رندرینگ صدای سه‌بعدی
- نمونه سازی واقعیت مجازی در حالت اجرا بازی در یونیتی با استفاده از موس و کلید‌های Alt و Cntrl

## 2-3- ساخت بازی در یونیتی

یونیتی یک موتور بازی چند سکویی است که برای ساخت بازی ویدئویی برای کامپیوترهای شخصی، کنسولهای بازی، تلفن‌های هوشمند یا اسمارت‌فونها و وب سایت‌ها استفاده می‌شود. یونیتی از جمله موتورهای بازی سازی است که از زبانهای برنامه‌نویسی معروفی نظریه سی‌شارپ، جاوا اسکریپت و زبان برنامه‌نویسی بو که یک زبان از خانواده زبان برنامه نویسی پایتون است پشتیبانی می‌کند. همچنین یونیتی یک موتور بازی ساز چند پلتفرم است، یعنی می‌تواند برای بسیاری از پلتفرم‌های موجود بازی را ایجاد کند.

---

<sup>1</sup> Binocular Rendering



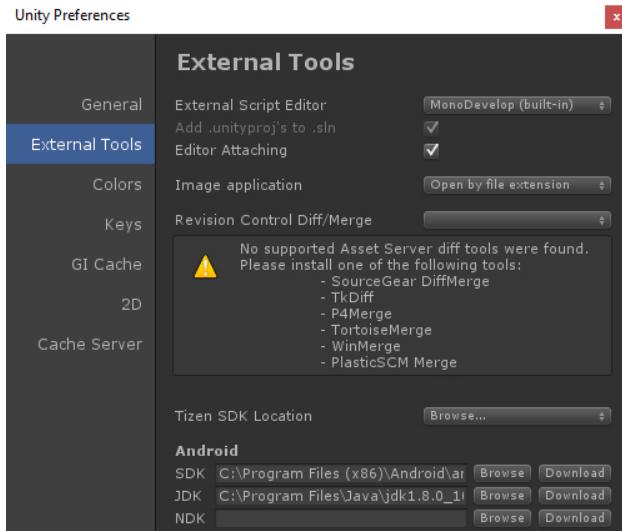
شکل (2-1) نمایی از موتور بازی ساز یونیتی

## 2-3-2- ایجاد پروژه جدید و انجام تنظیمات اولیه محیط بازی در یونیتی

برای شروع لازم است تا یک پروژه سه بعدی جدید در محیط یونیتی ایجاد کنیم آنگاه خواهیم دید که یک صحنه بازی به همراه یک دوربین اصلی<sup>1</sup> و منبع نوری<sup>2</sup> به عنوان تنها آبجکت‌های بازی ماد ایجاد شده‌اند. مسیر نصب کیت توسعه نرم افزار اندروید را که از قبل دانلود و نصب نموده در نرم‌افزار یونیتی از مسیر Edit > Preferences > External Tools تعیین می‌کنیم. البته این کار فقط یکبار لازم است انجام شود.

<sup>1</sup> Main Camera

<sup>2</sup> Directional Light



شکل (2-2) مشخص کرده مسیر نصب کیت توسعه نرم افزار اندروید در یونیتی

پیش از نصب کیت توسعه نرم افزار اندروید لازم است تا با توجه به 32 بیت یا 64 بیت بودن سیستم عامل خود نسخه<sup>1</sup> JDK مناسب را دریافت کنیم. پس از جستجوی کلمه JDK و هدایت به سایت Oracle می‌توان آنرا دانلود نمود. در اینجا ما با توجه به سیستم عامل خود که 64 بیتی می‌باشد نسخه را دانلود نموده و اقدام به نصب آن می‌کنیم که مراحل ساده‌ای در پیش دارد. حال نوبت به نصب SDK اندروید می‌رسد. سایت اندروید دولاپر<sup>2</sup> دو راه برای نصب در اختیار ما می‌دهد. می‌توانید اندروید استودیو را به طور کامل دانلود و نصب نموده و سپس از طریق آن اقدام به نصب موارد مورد نیاز کنیم ولی در صورتی که نیاز به برنامه نویسی اندروید نداشته باشیم این سایت امکان دانلود خطوط فرمات<sup>3</sup> را به تنها یی نیز در اختیار ما قرار می‌دهد که پس از دریافت یک فایل نصب کننده<sup>4</sup> می‌توان آن را به دانلود کیت توسعه نرم افزار نمود. در اینجا ما اندروید استودیو را دانلود و نصب نمودیم که نصب آن بسیار ساده بوده و همانند اکثر نرم افزارها نیازمند تعیین محل نصب و فشردن چندین دکمه Next می‌باشد. پس از نصب و اجرای اندروید استودیو از مسیر Tools > Android SDK Manager را اجرا کنیم. این همان فایلی است که در صورت دانلود اینساتلر برای ما نصب می‌شود و مدیریت SDK اندروید را بر عهده دارد و از طریق آن می‌توان تمامی فایل‌ها و API‌های مورد نیاز را دانلود نمود. باید توجه داشت که به علت تحریم آی‌پی ایران از سوی گوگل نیازمند تغییر آی‌پی مثلاً با استفاده از وی‌پی آن می‌باشیم تا بتوانیم

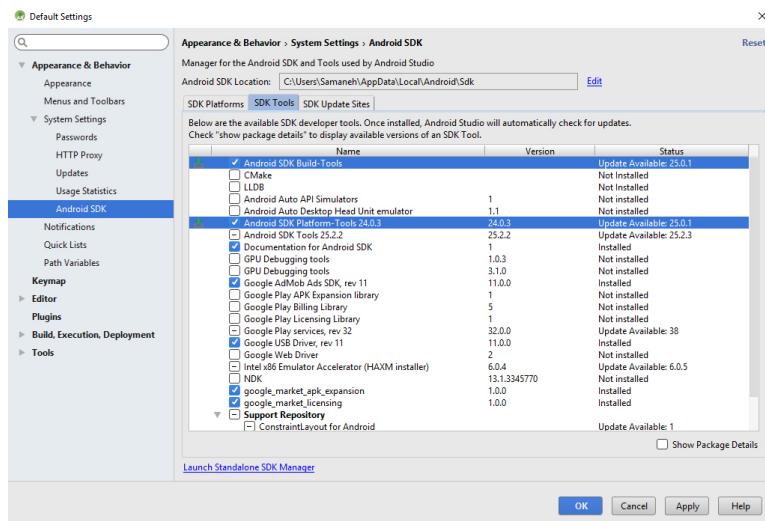
<sup>3</sup> Java Development Kit

<sup>4</sup> Developers.android.com

<sup>1</sup> Command Lines

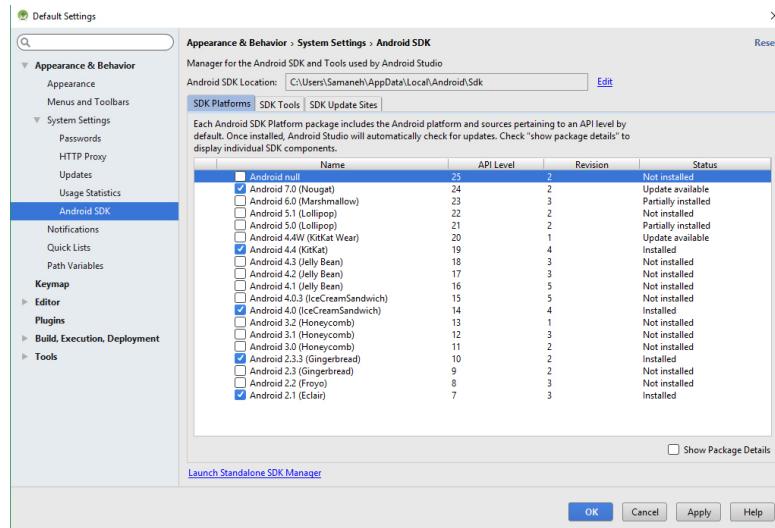
<sup>2</sup> Installer

فایل‌های SDK Manager را مشاهده و دانلود کنیم. ما به سه بخش اصلی از فایل‌های در دسترس نیازمندیم که عبارتند از SDK Tools، Android SDK Build Tools و API Android SDK Platform-Tools. اندروید. از تب SDK Tools دو مورد اول را دانلود می‌کنیم که معمولاً آخرین آپدیت موجود را به ما پیشنهاد می‌دهد اما در صورت نیاز می‌توان آپدیت‌های قدیمی‌تر را نیز دانلود کرد. برای مثال برخی از نسخه‌های یونیتی ممکن است با برخی ورژن‌های این فایل‌ها سازگاری نداشته باشند.



شکل (2-3) دریافت Build Tools و Android SDK Tools

حال لازم است تا از تب SDK Platforms آن دسته از API‌های مورد نیازمان را دانلود کنیم. برنامه‌نویس اندروید معمولاً از API‌های مرتبه پایین‌تر استفاده می‌کند که باعث می‌شود اپلیکیشن آنها قابلیت نصب روی تعداد گوشی‌های بیشتری را داشته باشد. با توجه به اینکه واقعیت مجازی به حداقل API مرتبه 19 نیاز دارد لازم است که آن را دانلود و نصب نماییم. البته یونیتی به هنگام خروجی گرفتن در صورت نصب نبوده آخرین ورژن اندروید خطایی مبنی بر عدم وجود آن می‌دهد و از شما می‌خواهد که آن را دانلود نمایید تا بتوان فایل خروجی نهایی را تولید کرد پس بهتر است که آن را نیز داشته باشیم.



شکل (2-4) دریافت API‌های مورد نیاز

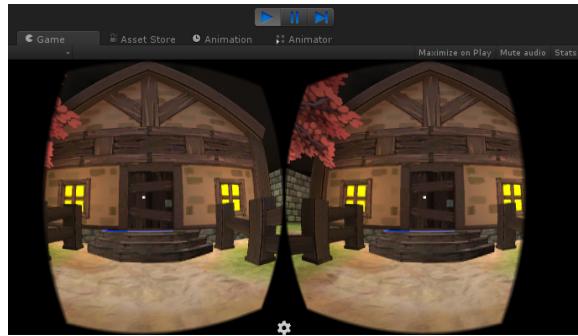
حالنوبت به وارد کردن SDK واقعیت مجازی گوگل برای یونیتی می‌رسد. از مسیر Assets>Import Package >Custom Package... کیت توسعه نرم افزار واقعیت مجازی گوگل برای یونیتی را که از قبل دانلود کرده‌ایم وارد پروژه می‌کنیم. این کیت نیز بر روی سایت گوگل دولاپرز<sup>1</sup> موجود بوده که برای ورود به آن به دلیل تحریم لازم است تا از وی‌پی‌اوز استفاده کنیم. روند ایجاد بازی واقعیت مجازی همانند بازی‌های سه بعدی معمولی است با این تفاوت که با افزودن این بسته به پروژه‌ماز از یکسری کدها و تنظیمات ویژه واقعیت مجازی مخصوص با عینک مقوایی گوگل نیز بهره می‌بریم تا بازی تبدیل به یک بازی واقعیت مجازی سازگار با گوشی هوشمندماز شود. همچنین قالبی که برای پروژه می‌خواهیم استفاده کنیم و حاوی آجکتها، کاراکترها و اینیمیشن‌های مورد نیاز است را به بازی از مسیر Assets > Import Package > Custom Package... اضافه می‌کنیم. ما برای این پروژه از یک بسته به نام خانه قرون وسطایی<sup>2</sup> که از فروشگاه یونیتی دانلود نموده به عنوان محیط اصلی بازی استفاده می‌کنیم. این بسته دارای پوشش‌های گوناگونی شامل آجکتها، کاراکترها، اینیمیشن‌ها و بسیاری موارد از پیش آمده شده دیگر است که می‌توان آنها را مطابق میل خود سفارشی‌سازی کرده و بر اساس نیاز هر کدام را که بخواهیم به بازی اضافه کرده و محیط بازی خود را بسازیم. تعدادی آجکت بازی خارج از بسته‌ی خانه قرون وسطایی مانند تعدادی کاراکتر از جمله کاراکتر Witch و Crawler، بسته‌های یونیتی حاوی اینیمیشن و همچنین بافت آتش برای ایجاد آتش نیز برای بازی‌ماز استفاده کرده‌ایم که به همین روش به پروژه‌ماز اضافه می‌کنیم.

<sup>1</sup> Developers.google.com

<sup>2</sup> Medieval House

### 2-3-3- تنظیمات دوربین

همانطور که گفتیم در هنگام ایجاد پروژه جدید، یک دوربین به بازی اضافه می‌شود. دوربین در واقع همان چیزی را نشان می‌دهد که کاربر در هنگام اجرای بازی می‌بیند اما این یک دوربین معمولی پرسپکتیو برای بازی‌های سه بعدی است و ما به یک دوربین مناسب با بازی واقعیت مجازی نیاز داریم که با نمایگشتر مقواپی گوگل سازگار باشد. این آبجکت در کیت توسعه نرم افزار واقعیت مجازی گوگل موجود است. کافی است تا در پنجره پروژه در نرم افزار یونتی از پوششی Prefabs درون پوششی GoogleVR نمایشگر واقعیت مجازی گوگل به نام GvrViewerMain را به بازی اضافه کنیم که دارای اسکریپتی به زبان سی شارپ به نام GvrViewer.cs است و با نمایشگر سربند ارتباط برقرار می‌کند. این امر سبب می‌شود تا دوربین معمولی بازی در یونیتی تبدیل به یک دوربین دوچشمی شود و رندرینگ برای هر چشم به صورت جداگانه صورت می‌گیرد.



شکل (5-2) تصویری از بازی در حالت اجرا هنگامی که از GvrViewerMain استفاده می‌شود

در نسخه‌های قدیمی این کیت توسعه نرم افزار لازم بود تا دوربین یونیتی را حذف کرده و دوربین واقعیت مجازی را که با نام CardboardMain بود از بسته به بازی اضافه کنیم. اما در نسخه‌های جدیدتر بدین صورت است که با اضافه کردن این آبجکت که به GvrViewerMain تغییر نام یافته، در حقیقت همان دوربین اصلی یونیتی ویژگی‌های لازم یک دوربین دوچشمی واقعیت مجازی را پیدا می‌کند و دیگر نیازی به حذف آن نیست. از آنجایی که زاویه دید برای بازی را اصطلاحاً تیراندازی اولاً شخص<sup>1</sup> در نظر گرفته‌ایم به این صورت که دوربین در حقیقت همان چشمان کاربر است، لازم است تا در حالی که دوربین اصلی یونیتی انتخاب شده است از پنجره Inspector و با کلیک بر روی دکمه Add Component یک مؤلفه‌ی جدید به آن اضافه کنیم که Character Controller نام دارد. مؤلفه‌ی Y قسمت Center آن را به 1- تغییر می‌دهیم تا به گونه‌ای به نظر برسد که دوربین به جای چشمان کاربر قرار دارد و ارتفاع که با Height نشان داده شده است را از 0 به -2- تغییر می‌دهیم چرا که نمی‌خواهیم سر کاربر یا همان دوربین بر روی زمین قرار بگیرد. برای اینکه بتوان تشخیص داد کاربر به کدام آبجکت بازی نگاه می‌کند باید یک مؤلفه‌ی دیگر نیز به دوربین بازی با نام

<sup>1</sup> First-person Shooter

Physics Raycaster اضافه کرد که یک اسکریپت سی شارپ می‌باشد و درون کیت توسعه نرم افزار نمایگشتر مقاوی واقعیت مجازی گوگل برای یونیتی قرار دارد. خط مستقیمی را تصور کنید که از دوربین خارج می‌شود و به هر آنچه دوربین با آن مواجه است برخورد می‌کند و بدین ترتیب امکان برقراری تعامل بین کاربر و هر یک از آبجکت‌های بازی که تعاملی هستند، بقرار می‌شود. به منظور اینکه کاربر بتواند راحت‌تر مرکز دوربین را تشخیص دهد و بداند دقیقاً به کدام نقطه نگاه می‌کند در کیت توسعه نرم افزار مقوای گوگل چیزی درست شیوه تار موئینی که معمولاً در دوربین‌های نقشه برداری وجود دارد در دسترس توسعه‌دهنگان بازی قرار گرفته که به صورت پیش‌فرض دایره‌ای شکل است و می‌توان رنگ، اندازه و شکل آنرا تغییر داد که GvrReticle نام دارد و از پوشه‌ی UI درون پوشه‌ی Prefabs از کیت می‌توان آنرا پیدا نمود و با اضافه کردن آن به دوربین اصلی نقطه‌ای درست در مرکز دوربین قرار می‌گیرد. به این ترتیب اگر کاربر قرار است به نقطه‌ای خاص از بازی نگاه کند این امر کار را راحت‌تر می‌کند. آخرین مؤلفه‌ای که می‌خواهیم به دوربین اصلی اضافه کنیم اسکریپتی است که خودمان کدنویسی می‌کنیم و امکان راه رفتن خودکار در محیط بازی را فراهم می‌کند. علت اضافه کردن این مؤلفه به دوربین اصلی این است که در واقع دوربین همان کاربری است که در حال اجرای بازی است و می‌خواهد در محیط بازی حرکت کند. راههای مختلف برای اضافه کردن این ویژگی به بازی وجود دارد از جمله نگاه کردن به یک شی برای مدت زمانی خاص که باعث فعال و غیرفعال کردن راه رفتن می‌شود، برای مثال می‌توان با ایجاد یک آبجکت مکعب از مسیر `<3D Object> <GameObject>` و قرار دادن آن بالای سر کاربر و در نهایت اضافه کردن این اسکریپت به عنوان یکی از مؤلفه‌های آبجکت مکعب، راه رفتن خودکار را از طریق نگاه کردن به یک شی انجام داد. روش دیگر تعیین زاویه دید است به این صورت که اگر کاربر به پایین نگاه کند و در نتیجه زاویه دید به یک زاویه خاص تعیین شده تغییر کند شروع به راه رفتن کرده و در صورت نگاه کردن به بالا متوقف شود. این اسکریپت به دوربین اصلی اضافه می‌شود چراکه زاویه دوربین برایمان مطرح است. روش سوم که در این پروژه انجام شده، فعال و غیرفعال کردن راه رفتن از طریق فشار دادن دکمه‌ای آهنربایی است که بر روی نمایشگر مقوای گوگل تعیین شده است. پس از اضافه کردن یک اسکریپت سی شارپ خالی، کد خود را به آن اضافه می‌کنیم و این اسکریپت نیز مانند روش قبل به عنوان یکی از مؤلفه‌های دوربین اصلی است. در ابتدا متغیرهای موردنیازمان را تعریف می‌کنیم. یک نوع داده Transform با نام VrCamera تعریف کرده که به مؤلفه Transform دوربین اشاره می‌کند. همچنین به دو متغیر دیگر یکی از نوع بولین برای تعیین فعال و غیرفعال بودن راه رفتن خودکار و دیگری از نوع اعشاری با مقدار اولیه ۱ که سرعت راه رفتن را تعیین می‌کند، نیاز داریم. متغیر سرعت و نیز متغیری که به دوربین اشاره دارد را از نوع عمومی تعریف می‌کنیم تا بتواند آنها در پنجره Inspector دسترسی داشت و در صورت نیاز مقادیر آنها را تغییر داد برای مثلاً می‌توان سرعت راه رفتن را به دلخواه تغییر دهیم و یا دوربین بازی را تعیین کنیم. همچنین به یک متغیر از نوع Character Controller نیاز داریم که در واقع به مؤلفه Character Controller دوربین اشاره می‌کند که همان کاربر است.

```

1 using UnityEngine;
2 using System.Collections;
3
4 public class vraw : MonoBehaviour {
5
6     public Transform vrCamera;
7     private bool walking = false;
8     public float speed = 1.0f;
9     private CharacterController cc;
10

```

شکل (6-2) متغیرهای مورد نیاز در اسکریپت راه رفتن خودکار

در تابع `Start()` که به صورت پیشفرض در هنگام ایجاد اسکریپت خالی به آن اضافه و در زمان شروع بازی فراخوانی می‌شود، مؤلفه‌ی `Character Controller` مربوط به آبجکتی که این اسکریپت به آن وصل شده است را می‌گیریم. به این ترتیب تعیین می‌کنیم که چه آبجکتی می‌خواهد حرکت کند که در اینجا دوربین بازی یا همان کاربر است.

```

void Start () {
    cc = GetComponent<CharacterController> ();
}

}

```

شکل (7-2) تابع شروع در اسکریپت راه رفتن

در تابع `Update()` که آن هم مانند تابع شروع به صورت پیشفرض با ایجاد اسکریپت ایجاد و در هر فریم از بازی فراخوانی می‌شود، لازم است تا ابتدا چک کنیم که اگر ورودی آهنربایی نمایشگر مقوایی فشار داده شده است مقدار متغیر بولین تغییر کند، به این معنا که اگر در حال راه رفتن هستیم متوقف و اگر متوقف هستیم شروع به راه رفتن کنیم. حال باید مشخص کنیم که اگر مقدار متغیر بولین صحیح است به این معنا که راه رفتن خودکار باید صورت گیرد، راه رفتن در چه جهتی و با چه سرعتی انجام گیرد. پس ابتدا جهت مؤلفه‌ی `Transform` برای متغیری که که از نوع عمومی و با نام `VrCamera` تعریف کرده‌ایم را می‌گیریم و آنرا به آبجکت جاری که اسکریپت به آن وصل شده می‌دهیم. سپس تابعی با نام `SimpleMove()` که یک حرکت ساده در جهت مشخص شده و با سرعت تعیین شده انجام می‌دهد برای کنترلر کاراکتر که همان متغیر `cc` است صدا می‌زنیم.

```

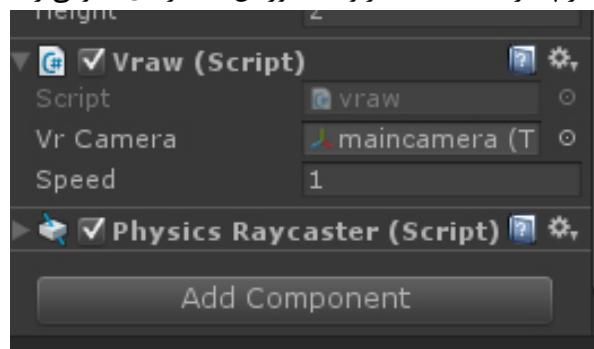
// Update is called once per frame
void Update () {
    if (GvrViewer.Instance.Triggered) {
        walking = !walking;
    }

    if (walking) {
        Vector3 direction = vrCamera.TransformDirection (Vector3.forward);
        cc.SimpleMove (direction * speed);
    }
}

```

شکل (2-8) تابع بروزرسانی در اسکریپت راه رفتن خودکار

پس از ذخیره اسکریپت در پنجره Inspector مربوط به دوربین، متغیرهای عمومی را مشخص می‌کنیم.



شکل (2-9) تنظیمات اسکریپت راه رفتن خودکار در پنجره Inspector

#### 2-3-4-آجکت‌های تعاملی در بازی

حال که به دوربین اصلی مؤلفه‌ی Physics Raycaster را اضافه کرده‌ایم لازم است تا از شناسایی آجکت‌های تعاملی موجود در صحنه‌ی بازی توسط آذ اطمینان حاصل کنیم. برای انجام این کار باید به آجکت‌های تعاملی مؤلفه‌ای به نام Collider اضافه کنیم. هنگامی که یک آجکت را در یونیتی می‌سازیم، آنها به صورت پیش فرض به همراه آذ ایجاد می‌شوند که اطراف شکل آجکت را فرا می‌گیرد و هر زمان‌چیزی آذ را لمس کرد در حقیقت آجکت را لمس کرده است. به عبارتی این مؤلفه به اشیا بازی همانند واقعیت موجودیت می‌بخشد در غیر این صورت سایر اشیا از آذ عبور خواهند کرد و برخوردي صورت نمی‌گیرد. اما اگر آجکتی خارج از محیط یونیتی مثلاً توسط نرم‌افزارهای اندیمیشن سازی ساخته شود و به یونیتی وارد کنیم دارای این مؤلفه نمی‌باشند و لازم است که به آنها اضافه شود. برای اضافه کردن این مؤلفه به یک آجکت در حالی که آذ آجکت انتخاب شده است از پنجره‌ی Inspector بر روی دکمه‌ی Add Component کلیک کرده و پس از جستجوی آذ را به آجکت اضافه می‌کنیم. نوع این مؤلفه را نیز بسته به شکل

کاراکتر انتخاب می‌کنیم، برای مثال در این پروژه برای کاراکترهای تعاملی بازی از نوع جعبه‌ای استفاده شده است.



شکل (2-10) مؤلفه Collider به صورت خطوط سبز رنگ اطراف آبجکت مشخص می‌شود

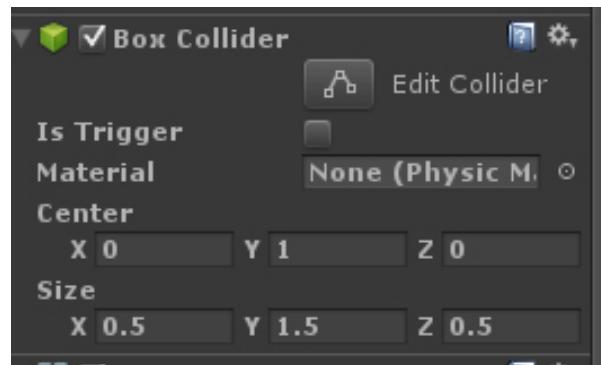
اگر به تنظیمات این مؤلفه نگاه کنیم خواهیم دید که مقادیری وجود دارند که می‌توان آنها را تغییر داد. البته مقادیر قابل تنظیم این مؤلفه برای انواع مختلف آذ فرقه می‌کند، برای مثال نوع کپسولی دارای مقادیر نقطه مرکزی<sup>۱</sup> است که Collider را اطراف آبجکت حرکت می‌دهد و برای زمانی که می‌خواهیم تنها بخشی از آبجکت و نه همه‌ی آذ تعاملی باشد بسیار سودمند است. مقادار بعدی شعاع<sup>۲</sup> است که برای کوچک و بزرگ کردن این مؤلفه استفاده می‌شود و باعث کاهش یا افزایش فضای بافر اطراف آبجکت می‌گردد. مقادار سوم ارتفاع<sup>۳</sup> است که مؤلفه را بلندتر یا کوتاه‌تر می‌کند و فضای بافر را بر روی محورها تغییر می‌دهد. نوع جعبه‌ای مقادیر مرکز و اندازه را دارا است که مقدار هر یک را می‌توان بر روی سه محور X، Y و Z تغییر داد.

---

<sup>1</sup> The Center Point

<sup>2</sup> The Radius

<sup>3</sup> The Height



شکل (2-11) تنظیمات مؤلفهی Collider برای یک آبجکت

تعامل میان کاربر و آبجکتها باری در پلتفرم واقعیت مجازی گوگل که از نمایشگر مقواپی آذ استفاده می‌کنیم به دو طریق امکان‌پذیر است، یا از طریق خیره نگاه کردن<sup>4</sup> به آذ شی که در آذ صورت پاسخی که بعد از نگاه کردن به شی باید صورت بگیرد انجام می‌شود. روش دوم با استفاده از دکمه‌ی آهنربایی نمایشگر مقواپی گوگل است، بدین صورت که در حال نگاه کردن به آبجکت تعاملی مورد نظر در بازی در صورت فشار دادن دکمه‌ی آهنربایی عملی که به منظور برقراری تعامل در نظر گرفته‌ایم انجام می‌شود. در اینجا چون از دکمه‌ی آهنربایی برای راه رفتن خودکار استفاده شده است، تعامل کاربر با آبجکتهای دروغ بازی از طریق خیره نگاه کردن به آذ آبجکتها صورت می‌گیرد. Raycaster از دوربین به آذ شی برخورد میکند که در حقیقت به مؤلفهی Collider آذ برخورد کرده و در نتیجه‌ی آذ عملی که می‌خواهیم اتفاق بیفتند، انجام می‌گیرد. این عمل می‌تواند هر چیزی باشد مانند اجرای یک انیمیشن، پخش صدا، جابه‌جایی یک کاراکتر و غیره.

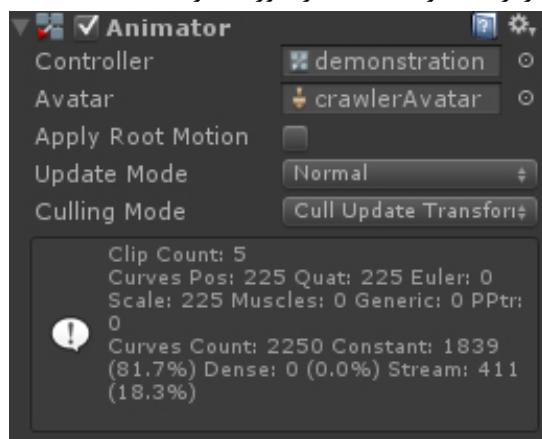
### 2-3-5- اسکریپت تعامل

می‌خواهیم زمانی که کاربر به آبجکتی خاص خیره نگاه کرد اتفاقی صورت بگیرد. می‌توان شمار گوناگونی از پاسخ‌هایی که به آبجکتهای تعاملی داده می‌شود در نظر گرفت. تغییر جنس یک آبجکت که در نتیجه‌اش رنگ آذ تغییر می‌کند، تغییر آبجکت به یک آبجکت دیگر، تغییر پس زمینه صحنه یا سایر آبجکتهای دیگر بازی و بسیار موارد دیگر.

در اینجا دو کاراکتر Cauldron و Crawler را به عنوان آبجکتهای تعاملی در نظر گرفته‌ایم و برای اینکه هرگاه کاربر به آنها نگاه کرد پاسخی صورت گیرد به هریک از این آبجکتها اسکریپتی به عنوان مؤلفهی آنها اضافه می‌کنیم و این اسکریپت را مطابق با پاسخ تعامل کدنویسی می‌کنیم. در ادامه به بررسی هریک به صورت جداگانه می‌پردازیم.

<sup>4</sup> Gaze

برای کاراکتر Crawler، تعدادی اینیمیشن در نظر گرفته‌ایم که با هربار نگاه کردن کاربر به کاراکتر مورد نظر یکی از این اینیمیشن‌ها به ترتیبی که تعیین می‌کنیم اجرا شده و در نهایت پس از اجرای اینیمیشن آخر دوباره به اولین اینیمیشن بازمی‌گردد و در حقیقت اجرای اینیمیشن‌ها مانند یک حلقه عمل می‌کند. بدین منظور ابتدا کاراکتر مورد نظرمان باید دارای مؤلفه‌ای به نام Animator باشد که آنرا از پنجره‌ی Inspector می‌توان به آبجکت‌مان اضافه کرد. سپس باید کنترلر کاراکتر را که معمولاً همراه کاراکتر درون پوششی آن قرار دارد به قسمت Controller در بخش تنظیمات از این مؤلفه اضافه کنیم و نیز تیک گزینه‌ی Apply Root Motion را برداریم چراکه این گزینه سبب می‌شود تا اینیمیشن کاراکتر را در فضای واقعی دنیای بازی حرکت دهد ولی ما می‌خواهیم با هربار نگاه کردن به آذینکار صورت گیرد.

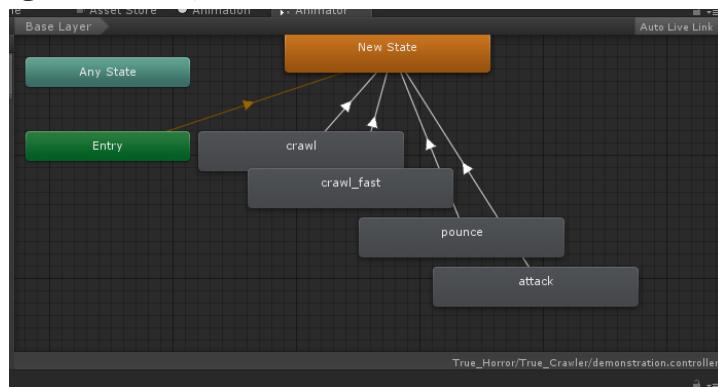


شکل (2-12) تنظیمات مؤلفه‌ی Animator برای کاراکتر Crawler

آنگاه در پنجره‌ی Animator که می‌توان آنرا از مسیر Window > Animator به محیط یونیتی اضافه کرد لازم است تا برای اینیمیشن‌ها یک ماشین حالت تعریف کنیم. پس کلیپ‌های اینیمیشن مورد نیاز را به درون این پنجره از طریق درگ کردن وارد می‌کنیم. سپس با کلیک راست کردن بر روی صفحه و انتخاب گزینه‌ی Create State یک حالت خالی ایجاد می‌کنیم و بر روی حالت خالی ایجاد شده راست کلیک کرده و با انتخاب Set as Layer Default State آنرا به عنوان حالت پیشفرض در نظر می‌گیریم که رنگ آن به نارنجی تغییر می‌کند و به این معناست که این حالت به عنوان وضعیتی است که دائمًا در بازی بر روی کاراکتر نمایش داده می‌شود که در این وضعیت کاراکتر ثابت است چراکه حالت خالی انتخاب شده است. در صورتی که به جای یک حالت خالی یکی از اینیمیشن‌ها را به عنوان حالت پیشفرض تعیین می‌کردیم آذینکار اینیمیشن دائمًا در طوا بازی اجرا می‌شد. حالتی که به نام Entry مشخص شده و با رنگ سبز نمایش داده می‌شود به معنی ورودی این ماشین حالت است که ما آنرا به حالت پیشفرض وصل می‌کنیم. حال می‌خواهیم جایه‌جایی از یک حالت به حالت دیگر<sup>1</sup> ایجاد کنیم که باعث اجرای یک اینیمیشن شده و دوباره به حالت پیشفرض بازگردد به

<sup>1</sup> Transition

عبارت دیگر هرگاه کاربر به کاراکتر نگاه کرد یکی از انیمیشن‌ها اجرا و پس از اتمام آن دوباره به حالت پیشفرض که ثابت بودن کاراکتر است بازگردد. برای این منظور بر روی تک تک انیمیشن‌ها راست کلیک کرده و با انتخاب Make Transition یک انتقال از انیمیشن مورد نظر به حالت پیشفرض ایجاد می‌کنیم.



شکل (13-2) ایجاد ماشین حالت در Animator یونیتی

حال نوبت به اسکریپت تعامل می‌رسد که به عنوان مؤلفه‌ای از این کاراکتر به آن وصل می‌شود. در کدام‌به یک متغیر عمومی از نوع Animator که به این مؤلفه از کاراکتر اشاره می‌کند و همچنین به یک متغیر صحیح که شماره‌ی انیمیشنی که باید اجرا شود را تعیین می‌کند نیاز داریم.

```

1 using UnityEngine;
2 using System.Collections;
3
4 public class CrawlerScript : MonoBehaviour {
5
6
7     public Animator cAnim;
8     private int curranim= 0;
9     // Use this for initialization
  
```

شکل (14-2) متغیرهای مورد نیاز در اسکریپت تعامل با Crawler

حال نیاز داریم تا یک تابع جدید تعریف کنیم که هر بار کاربر به کاراکتر نگاه می‌کند اجرا شود و با توجه به مقدار متغیری که شماره‌ی انیمیشن جاری را تعیین می‌کند انیمیشن مورد نظر را اجرا و شمارنده را یک واحد افزایش دهد.

```

public void StareAtCrawler(){
    switch (curanim) {
        case 0:
            cAnim = GetComponent<Animator> ();
            cAnim.Play ("crawl", -1, 0f);
            curanim++;
            break;
        case 1:
            cAnim = GetComponent<Animator> ();
            cAnim.Play ("crawl_fast", -1, 0f);
            curanim++;
            break;
        case 2:
            cAnim = GetComponent<Animator> ();
            cAnim.Play ("pounce", -1, 0f);
            curanim++;
            break;
        case 3:
            cAnim = GetComponent<Animator> ();
            cAnim.Play ("attack", -1, 0f);
            curanim = 0;
            break;
    }
}

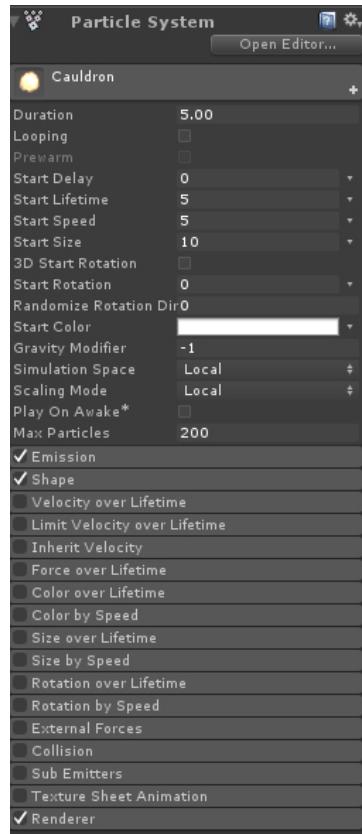
```

شکل (2-15) تابع پاسخ به تعامل با کاراکتر Crawler

برای آبجکت Cauldron، قصد داریم که هر بار کاربر به آذنگاه کرد آتشی از آذ خارج شود. برای ایجاد آتش از سیستم ذره<sup>1</sup> یونیتی استفاده می‌کنیم. ذرات می‌توانند مبنای یک آتش باشند یا افکت‌های درخشان افسوز یک جادوگر را ایجاد کنند. همچنین برای ایجاد دود، گردخاک انفجار و حتی رنگین کماز استفاده می‌شوند. این ذرات بسیار کوچک، تصاویر یا مشاهی بسیار ساده‌ای هستند که به بیرون منتشر شده و سپس در طول عمر خود توسط سیستم ذره دستکاری می‌شوند. یک سیستم ذره عموماً ذرات را در مکانهای تصادفی درون یک مکان از پیش تعریف شده منتشر می‌کند که می‌تواند مانند یک سکه یا کره شکل داده شود. یکی از مزیت‌های این سیستم در یونیتی این است که می‌توان آنرا به عنوان مؤلفه‌ای از یک آبجکت بازی به آذ اضافه کرد.

---

<sup>1</sup> Particle System



شکل (2-16) بخش های مختلف تنظیمات سیستم ذره

ماژوا اصلی هر سیستم ذره شامل تنظیمات مختلفی از جمله، مدت<sup>۱</sup> که طول مدت زمان اجرای سیستم ذره بر حسب ثانیه است و ما آن را برابر ۵ قرار دادیم، حلقه زنی<sup>۲</sup> که ذرات را تا زمانی که سیستم ذره متوقف شود تکرار می کند و چرخه زمانی که عمر سیستم به پایان رسید مجددآ شروع می شود و ما آن را غیرفعال می کنیم، تأخیر راه اندازی<sup>۳</sup> که ثانیه های قبل از شروع انتشار است و به صورت پیش فرض صفر است، شروع دوره زندگی<sup>۴</sup> که در واقع عمر ذرات بر حسب ثانیه است و ذره بعد از این زمان نابود می شود که برابر ۵ قرار می دهیم، سرعت راه اندازی<sup>۵</sup> که سرعت اولیه ذرات است و هرچه بیشتر باشد ذرات بیشتر به بیرون منتشر

---

<sup>1</sup> Duration

<sup>2</sup> Looping

<sup>3</sup> Start Delay

<sup>4</sup> Start Lifetime

<sup>5</sup> Start Speed

می‌شوند و برابر 5 قرار می‌دهیم. پس از انجام این تنظیمات مشاهده می‌کنیم که آتش تقریباً حالت خوبی دارد اما شبیه به دود است و نیاز به دستکاری بیشتری دارد پس نیاز به تنظیمات دیگری داریم. مازو<sup>6</sup> اصلی تنظیمات اضافه دیگری نیز دارد از جمله، اندازه راه اندازی<sup>1</sup> که اندازه ابتدایی ذرات در شروع است و آنرا برابر 10 قرار می‌دهیم، چرخش راه اندازی<sup>2</sup> که زاویه اولیه چرخش ذرات است و آنرا برابر صفر قرار می‌دهیم در نتیجه ذرات گرد هستند، رنگ راه اندازی<sup>3</sup> که آنرا سفید خالص رها می‌کنیم چراکه می‌خواهیم با استفاده از بافت آتشی که از قبل دانلود کرده‌ایم ظاهر آنرا شبیه به آتش واقعی کنیم، تعدیل کننده جاذبه<sup>4</sup> که اگر برابر صفر باشد به معنی خاموش بودن جاذبه است و ما آنرا برابر 1- قرار می‌دهیم تا آتش به سمت بالا حرکت کند و اگر برابر 1 باشد همانند آبشار به پایین سقوط می‌کند، اجرا در زمان شروع<sup>5</sup> که اگر تیک این گزینه را بزنیم افکت آتش به محض شروع بازی اجرا می‌شود در غیر اینصورت باید به صورت دستی آنرا فعال کنیم پس ما تیک گزینه را برمی‌داریم چراکه می‌خواهیم در زمان نگاه کردن کاربر به آبجکت فعال شود، مازکزیم ذرات<sup>6</sup> که مازکزیم مقدار ذراتی که سیستم ذره اجازه دارد در هر زمان داشته باشد را مشخص می‌کند و ما آنرا برابر 200 قرار می‌دهیم. حال نوبت به مازو<sup>7</sup> نشر<sup>7</sup> میرسد که سرعت حرکت تعداد ذرات منتشر شده در هر ثانیه (زمان) را مشخص می‌کند و ما آنرا برابر 50 قرار می‌دهیم. همچنین می‌توان زمان را به فاصله تغییر داد. حالا می‌خواهیم یک بافت آتش به آزاد اضافه کنیم تا طبیعی جلوه کند پس از قسمت مازو<sup>8</sup> رندر<sup>8</sup>، بافت آتش مورد نظرمان را به قسمت Material می‌کشیم. آخرین تنظیمی که لازم است انجام دهیم تغییر شکل سیستم ذره است. مازو<sup>9</sup> شکل<sup>9</sup> همانطور که از نامش پیداست شکل و رفتار سیستم ذره در آن شکل را کنترل می‌کند.

می‌توان شکل‌های مختلفی را انتخاب کرد که هر کدام تنظیمات متفاوتی دارد. ما از شکل کره با شعاع 4 استفاده کردیم و در نهایت به نتیجه‌ی دلخواه رسیدیم.

---

<sup>6</sup> Start Size

<sup>7</sup> Start Rotation

<sup>8</sup> Start Color

<sup>9</sup> Gravity Modifier

<sup>10</sup> Play on Awake

<sup>11</sup> Max Particles

<sup>1</sup> Emission

<sup>2</sup> Renderer

<sup>3</sup> Shape



شکل (2-17) ایجاد افکت آتش با استفاده از سیستم ذره یونیتی

تا اینجا افکت آتش را با استفاده از سیستم ذره یونیتی ایجاد کردیم و اکنون نوبت به ایجاد اسکریپت تعامل برای استفاده از آن می‌رسد. پس از اضافه کردن یک اسکریپت خالی به عنوان مؤلفه‌ای از آبجکت به آن تابعی بسیار ساده اضافه می‌کنیم. این تابع مؤلفه‌ی سیستم ذره آبجکت جاری که اسکریپت به آن وصل شده است را می‌گیرد و آنرا اجرا می‌کند.

```

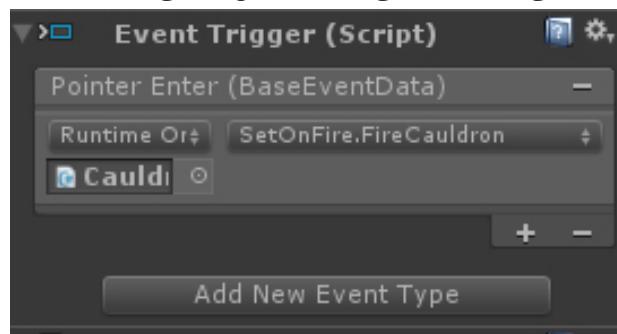
1 using UnityEngine;
2 using System.Collections;
3
4 public class SetOnFire : MonoBehaviour {
5
6
7     // Use this for initialization
8     void Start () {
9
10    }
11
12     // Update is called once per frame
13     void Update () {
14
15    }
16
17     public void FireCauldron(){
18         ParticleSystem ps = GetComponent<ParticleSystem> ();
19         ps.Play ();
20    }
21 }
22

```

شکل (2-18) اسکریپت تعامل برای آبجکت Cauldron

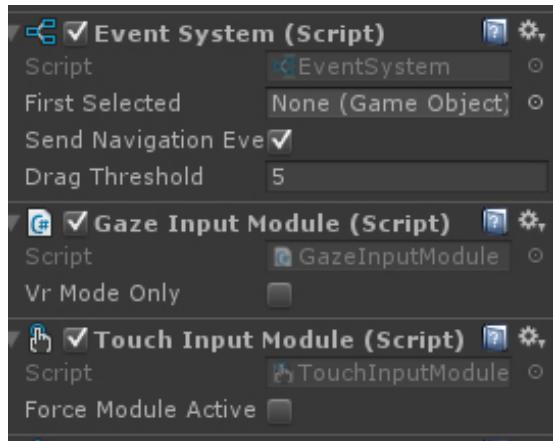
## 2-3-6- چگونه تعامل اتفاق می‌افتد

در قسمت قبل به آبجکت‌های تعاملی اسکریپتی اضافه کردیم که به عنوان پاسخ تعامل بود. اما این اسکریپت به خودی خود کاری انجام نمی‌دهد. برای اینکه این عملکرد کار کند لازم است تا یک مؤلفه‌ی نهایی به آبجکت‌های تعاملی اضافه کنیم که به آن Event Trigger گفته می‌شود. پس از اضافه کردن آن گزینه‌ای به نام Add New Event Type در این مؤلفه مشاهده می‌کنیم که با کلیک بر روی آن و انتخاب Pointer Enter تعیین می‌کنیم که با نگاه کردن به آن و در واقع قرار گرفتن اشاره‌گر بر روی آبجکت مورد نظر پاسخ به تعامل که همان اسکریپت است اتفاق بیفتند. حال باید آبجکت مورد نظر را از پنجره‌ی Hierarchy بشیم و بر روی قسمتی که در زیر نوشته‌ی RunTime On قرار گرفته است بیاندازیم. آنگاه بر روی قسمت No Function کلیک کرده و تابعی که به عنوان پاسخ در اسکریپت تعامل نوشته‌ایم را انتخاب می‌کنیم. باید توجه داشت که توابع در صورتی نمایش داده می‌شوند که از نوع عمومی تعریف شده باشند.



شکل (2-19) اضافه کردن مؤلفه Event Trigger به آبجکت تعاملی

اکنون تقریباً به نقطه‌ای که Event Trigger اجرا خواهد شد رسیده‌ایم! قبل از اینکه صحنه‌ی یونیتی بداند که باید دنبال آنها بگردد لازم است تا یک Event System از مسیر GameObject > UI > Event System به پروژه‌مان اضافه کنیم. درون آن می‌توان تعیین کرد که به دنبال رخداد نگاه کردن توسعه نرم‌افزار مقواه گوگل بگردد. بنابراین لازم است تا مؤلفه‌ای به نام Gaze Input Module به آن اضافه کنیم که این مؤلفه توسعه کیت گوگل ارائه می‌شود. توجه داشته باشید که این مؤلفه باید بالای مؤلفه‌ی Standalone Input Module قرار بگیرد در غیر این صورت کار نخواهد کرد. در صورتی که تعامل میان کاربر و اشیا بازی از طریق دکمه‌ی آهنربایی نمایشگر صورت می‌گرفت مؤلفه‌ی Touch Input Module را اضافه می‌کردیم که ما آنرا از قبل اضافه کرده‌ایم چراکه برای راه رفتن خودکار به آذنیاز داریم.



شکل (2-20) مؤلفه های اضافه شده به آبجکت Event System

### 2-3-7- خروجی گرفتن و اجرای بازی بر روی هدست واقعیت مجازی

برای تست بازی با نمایشگر مقواپی گوگل نیاز داریم تا بازی را دروز یونیتی برای یکی از سیستم عامل های اندروید یا IOS بسازیم. برای شروع پروسه ساخت بازی می تواند به تنظیمات ساخت بازی از مسیر ... File > Build Setting رفته و از آنجا پلتفرم مورد نظر خود را انتخاب نماییم. پس از انتخاب اندروید و کلیک بر روی دکمهی Switch Platform لازم است تا بر روی Player Setting کلیک کرده و تنظیمات بیشتری را برای بازی انجام دهیم. در فیلد Other Settings باید Bundle Identifier را مشخص کنیم (برای مثال Com.Samaneh.Spooky) و همچنین برای Android 4.4 Minimum API level باید Resolution را انتخاب کنیم که حداقل ورژن اندرویدی است که واقعیت مجازی را پشتیبانی می کند. در مرحله بعد از فیلد Landscape Left and Presentation را انتخاب کنیم و بدین ترتیب تمامی تنظیمات ساخت بازی واقعیت مجازی انجام می شود و در نهایت می تواند بر روی دکمهی Build کلیک کرده و خروجی اندروید مورد نظر را بدست بیاوریم. سپس فایل APK ایجاد شده را بر روی گوشی نصب و با اجرای آن و نیز قرار دادن گوشی هوشمند در نمایشگر مقواپی گوگل می تواند محیط واقعیت مجازی را مشاهده نمود.



شکل (2-21) اجرای بازی در نمایشگر واقعیت مجازی گوگل

# **فصل سوم**

## **ارتباط دستگاه‌های اندروید از طریق WiFiP2P**

### 3-1- مقدمه

وای-فای جفت به جفت<sup>1</sup> یا P2P به دستگاه‌های اندرویدی ورژن 4 API مرتبه 14) و یا بالاتر که دارای سخت افزار مناسب هستند امکان اتصال به یکدیگر را به صورت مستقیم از طریق وای-فای و بدون نیاز به یک نقطه دستیابی میانی<sup>2</sup> فراهم می‌کند. با استفاده از این API‌ها می‌توانید دستگاه‌های دیگر را در صورت پشتیبانی از وای-فای P2P شناسایی کرده و به آنها متصل شوید، آنگاه بوسیله‌ی یک ارتباط بسیار سریع تر و با فاصله‌ای بسیار دورتر نسبت به بلوتوث با دیگر دستگاه‌ها ارتباط برقرار کنید. این مسئله برای اپلیکیشن‌هایی که نیاز به تبادل داده می‌آزاد کاربران دارد، مانند بازی‌های چند کاربره و یا اپلیکیشن ارسال تصویر بسیار مناسب است.

### P2P-های وای-فای API-3-2

این API‌ها شامل متدهایی در کلاس WiFiP2pManager است که به شما اجازه‌ی شناسایی، درخواست اتصال و برقراری ارتباط با سایر دستگاه‌ها را می‌دهد. همچنین دارای شنونده‌ها<sup>3</sup> بی است که موفقیت یا عدم موفقیت فراخوانی متدهای کلاس WiFiP2pManager را اطلاع می‌دهند. وقتی این متدها فراخوانی می‌شوند هر یک می‌تواند یک شنونده معین را به عنوان یک پارامتر قبول کند. علاوه بر آن این API‌ها دارای Intent هایی است که شما را از رخدادهای ویژه شناسایی شده توسط بدنه وای-فای P2P نظیر یک ارتباط قطع شده یا دستگاه جدید شناسایی شده مطلع می‌کند.

### WifiP2pManager- کلاس 3-3

این کلاس متدهایی را عرضه می‌کند که به ما اجازه تعامل با سخت افزار وای-فای را دستگاه‌مان را می‌دهد تا کارهایی را که برای شناسایی سایر دستگاه‌ها و اتصال به آنها لازم است انجام دهیم. در جداول زیر به برخی از این متدها، شنونده‌ها و Intent‌ها اشاره شده است.

<sup>1</sup> WiFi peer-to-peer

<sup>2</sup> Intermediate Access Point

<sup>3</sup> Listeners

**جدول (3-1) متدهای وای-فای P2P**

Initialize()	رجیستر کردن اپلیکیشن با بندنه وای-فای، این متدهای قبیل از هر متدهای فراخوانی شود
Connect()	یک اتصال جفت به جفت با دستگاهی دارای پیکربندی معین شروع می‌کند
Cancelconnect()	مذاکره‌های گروهی جفت به جفت در حال پیشرفت را کنسل می‌کند
RequestConnectInfo()	اطلاعات اتصال یک دستگاه را درخواست می‌کند
CreateGroup()	یک گروه جفت به جفت با دستگاه حاری به عنوان دارنده گروه ایجاد می‌کند
RemoveGroup()	گروه جفت به جفت حاری را حذف می‌کند
RequestGroupInfo()	اطلاعات گروه جفت به جفت را درخواست می‌کند
DiscoverPeers()	شناختی جفت را آغاز می‌کند
RequestPeers()	لیست حاری جفتهای شناختی شده را درخواست می‌کند

**جدول (3-2) شنووندهای وای-فای P2P**

واسطه شنونده	اعمال وابسته
WifiP2pManager.ActionListener	Connect(), CancelConnect(), CreateGroup(), RemoveGroup(), DiscoverPeers()
WifiP2pManager.ChannelListener	Initialize()
WifiP2pManager.ConnectionInfoListener	RequestConnectInfo()
WifiP2pManager.GroupInfoListener	RequestGroupInfo()
WifiP2pManager.PeerListListener	RequestPeers()

**جدول (3-3) P2P اینتents وای-فای**

WIFI_P2P_CONNECTION_CHANGED_ACTION	وقتی که حالت اتصال وای-فای دستگاه تغییر می‌کند، منتشر می‌شود
WIFI_P2P_PEERS_CHANGED_ACTION	وقتی که DiscoverPeers() را فراخوانی می‌کنید منتشر می‌شود. معمولاً این متدهای برای گرفتن لیست بروزرسانی شده جفتهای فراخوانی می‌کنید
WIFI_P2P_STATE_CHANGED_ACTION	وقتی که وای-فای P2P دستگاه فعال یا غیرفعال است منتشر می‌شود
WIFI_P2P_THIS_DEVICE_CHANGED_ACTION	وقتی که جزئیات یک دستگاه مانند نام، تغییر می‌کند منتشر می‌شود

### 3-4- ایجاد یک دریافت کننده انتشاری برای Intent‌های وای-فای P2P

یک دریافت کننده انتشاری<sup>1</sup> به ما اجازه می‌دهد تا Intent‌های منتشر شده از سیستم اندرویدی را دریافت کنیم بنابراین اپلیکیشن ما به رخدادهای مدنظر ما می‌تواند پاسخ دهد. برای ایجاد یک دریافت کننده انتشاری لازم است تا ابتدا یک کلاس که BroadcastReceiver را تعمیم می‌دهد ایجاد کنیم. برای ساختار کلاس تمایل داریم تا پارامترهایی را برای WifiP2pManager.Channel، WifiP2pManager و WifiP2pManager.ActionListener

<sup>1</sup> Broadcast Receiver

عملی که این دریافت کننده انتشاری در آذ رجیستر می شود داشته باشیم. این امر به دریافت کننده انتشاری این امکان را می دهد که به همان اندازه که می تواند به سخت افزار وای فای و یک کانال ارتباطی در صورت نیاز دسترسی داشته، آپدیت هایی را نیز به آذ عمل ارسال کند. سپس در دریافت کننده انتشاری Intent های مدنظرمان را در ( ) onReceiver( چک می کنیم برای مثلاً اگر دریافت کننده انتشاری یک WIFI\_P2P\_PEERS\_CHANGED\_ACTION دریافت کند، می تواند متده ( ) requestPeers( را فراخوانی کرده تا یک لیست از جفت های شناسایی شده جاری را دریافت کنیم. کد زیر نشان می دهد که چگونه یک نمونه دریافت کننده انتشاری ایجاد کنید. این دریافت کننده یک آبجکت WiFip2pManager و یک Activity به عنوان آر گومان می گیرد و از این دو کلاس برای اجرای مناسب اعمال مورد نیاز در زمان دریافت یک استفاده می کند.

```
/*
 * A BroadcastReceiver that notifies of important Wi-Fi p2p events.
 */
public class WiFiDirectBroadcastReceiver extends BroadcastReceiver {

    private WiFip2pManager mManager;
    private Channel mChannel;
    private MyWiFiActivity mActivity;

    public WiFiDirectBroadcastReceiver(WIFIp2pManager manager, Channel channel,
                                       MyWiFiActivity activity) {
        super();
        this.mManager = manager;
        this.mChannel = channel;
        this.mActivity = activity;
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if (WIFIp2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {
            // Check to see if Wi-Fi is enabled and notify appropriate activity
        } else if (WIFIp2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {
            // Call WiFip2pManager.requestPeers() to get a list of current peers
        } else if (WIFIp2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {
            // Respond to new connection or disconnections
        } else if (WIFIp2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION.equals(action)) {
            // Respond to this device's wifi state changing
        }
    }
}
```

شکل (3-1) ایجاد یک دریافت کننده انتشاری

### 3-3- ایجاد یک اپلیکیشن وای-فای P2P

پیش از استفاده از API های وای-فای P2P لازم است تا از دسترسی اپلیکیشن خود به سخت افزار و اینکه دستگاه از پروتکل وای-فای P2P پشتیبانی می کند اطمینان حاصل نماییم. اگر وای-فای P2P پشتیبانی می شود می توان یک نمونه از WiFip2pManager دریافت کرده ، دریافت کننده انتشاری خود را ایجاد و

رجیستر نموده و از API‌های واي-فای P2P استفاده کنیم.

### 3-5-1- تنظیمات اولیه

در مرحله اول لازم است تا در فایل Android Manifest پروژه اندرویدی خود در اندروبود استودیو برای استفاده از سخت افزار واي-فای روی دستگاه، درخواست اجازه نموده و اپلیکیشن خود را برای داشتن حداقل ورژن SDK صحیح اعلام کنیم.

```
<uses-sdk android:minSdkVersion="14" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

شکل (3-2) مرحله اول از تنظیمات اولیه اپلیکشن واي-فای P2P

در مرحله دوم لازم است تا چک کنیم که آیا واي-فای P2P روشن بوده و پشتیبانی می‌شود یا خیر. بهترین مکان برای چک کردن در دریافت کننده انتشاری به هنگام دریافت WIFI\_P2P\_STATE\_CHANGED\_ACTION است.

```
@Override
public void onReceive(Context context, Intent intent) {
    ...
    String action = intent.getAction();
    if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {
        int state = intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE, -1);
        if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) {
            // Wifi P2P is enabled
        } else {
            // Wi-Fi P2P is not enabled
        }
    }
    ...
}
```

شکل (3-3) مرحله دوم از تنظیمات اولیه اپلیکشن واي-فای P2P

در مرحله سوم لازم است تا در متده onCreate() از Activity برنامه خود، یک نمونه از WifiP2pManager را دریافت و اپلیکیشن خود را با فریم ورک واي-فای P2P بوسیله فراخوانی initialize() رجیستر کنیم. این متده یک WifiP2pManager.Channel را که برای اتصال اپلیکیشن به فریم ورک واي-فای

P2P استفاده می‌شود برای گرداند. همچنین باید یک نمونه از دریافت کننده انتشاری خود با آبجکت‌های WifiP2pManager و WifiP2pManager.Channel با یک مرجع به Activity ایجاد کنیم. این امر به دریافت کننده انتشاری اجازه می‌دهد تا رخدادهای جالب آگاه کرده و طبق آن آپدیت کند. همچنین به ما این امکان را می‌دهد تا در صورت نیاز وضعیت وای‌فای دستگاه را دستکاری کنیم.

```

WifiP2pManager mManager;
Channel mChannel;
BroadcastReceiver mReceiver;
...
@Override
protected void onCreate(Bundle savedInstanceState){
    ...
    mManager = (WifiP2pManager) getSystemService(Context.WIFI_P2P_SERVICE);
    mChannel = mManager.initialize(this, getMainLooper(), null);
    mReceiver = new WiFiDirectBroadcastReceiver(mManager, mChannel, this);
    ...
}

```

شکل (3-4) مرحله سوم از تنظیمات اولیه اپلیکشن وای‌فای P2P

در مرحله چهارم لازم است تا یک فیلتر Intent ایجاد کرده و همان Intent‌ها را به دریافت کننده انتشاری برای چک کردن اضافه کنیم.

```

IntentFilter mIntentFilter;
...
@Override
protected void onCreate(Bundle savedInstanceState){
    ...
    mIntentFilter = new IntentFilter();
    mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);
    mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);
    mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);
    mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);
    ...
}

```

شکل (3-5) مرحله چهارم از تنظیمات اولیه اپلیکشن وای‌فای P2P

در مرحله پنجم لازم است تا دریافت کننده انتشاری را در متد () از onResume() رجیستر و در متد () از onPause() از رجیستری خارج کنیم.

```

/* register the broadcast receiver with the intent values to be matched */
@Override
protected void onResume() {
    super.onResume();
    registerReceiver(mReceiver, mIntentFilter);
}
/* unregister the broadcast receiver */
@Override
protected void onPause() {
    super.onPause();
    unregisterReceiver(mReceiver);
}

```

شکل (3-6) مرحله پنجم از تنظیمات اولیه اپلیکشن وای-فای P2P

هنگامی که یک WifiP2pManager.Channel می‌گیریم و یک دریافت کننده انتشاری تنظیم می‌کنیم، اپلیکیشن ما قادر است تا متد های وای-فای P2P را فراخوانی کرده و Intent های وای-فای P2P را دریافت کند. حال می‌توان اپلیکیشن خود را پیاده سازی کرده و از مشخصه های وای-فای P2P بوسیله فراخوانی متد های موجود در WifiP2pManager استفاده نمود. در بخش های بعدی به بررسی اعمال رایجی نظری شناسایی و اتصال به سایر دستگاه ها می‌پردازیم.

### 3-5-2- شناسایی جفت‌ها

برای شناسایی جفت‌های آمده اتصال لازم است تا discoverPeers() را برای شناسایی جفت‌هایی که در محدوده قرار دارند فراخوانی کنیم. فراخوانی این تابع آسنکرون بوده و موفقیت یا عدم موفقیت در برقراری ارتباط را با onSuccess() و onFailure() به اپلیکیشن اطلاع می‌دهد. اگر یک ایجاد کرده‌اید، متد onSuccess() فقط شما را از موفقیت آمیز بودن پروسه شناسایی مطلع می‌کند و هیچ اطلاعات دیگری را درباره جفت‌های شناسایی شده عرضه نمی‌کند.

```

mManager.discoverPeers(channel, new WifiP2pManager.ActionListener() {
    @Override
    public void onSuccess() {
        ...
    }

    @Override
    public void onFailure(int reasonCode) {
        ...
    }
});

```

شکل (3-7) فراخوانی تابع discoverPeers()

اگر پروسه شناسایی موفقیت آمیز بوده و جفت‌هایی شناسایی شود، سیستم WIFI\_P2P\_PEERS\_CHANGED\_ACTION را انتشار می‌دهد که می‌توانیم به گوش دادن دریافت‌کننده انتشاری برای گرفتن لیست جفت‌ها بپردازیم. وقتی اپلیکیشن ما این Intent را دریافت می‌کند، می‌توانیم یک لیست از جفت‌های شناسایی شده را با requestPeers() درخواست کنیم.

```

PeerListListener myPeerListListener;
...
if (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {

    // request available peers from the wifi p2p manager. This is an
    // asynchronous call and the calling activity is notified with a
    // callback on PeerListListener.onPeersAvailable()
    if (mManager != null) {
        mManager.requestPeers(mChannel, myPeerListListener);
    }
}

```

شکل (3-8) تنظیم WIFI\_P2P\_PEERS\_CHANGED\_ACTION

متدهای requestPeers() نیز آسنکرون بوده و هنگامی که لیستی از جفت‌ها در دسترس باشد Activity را با استفاده از onPeersAvailable() که در واسطه WifiP2pManager.PeerListListener است مطلع می‌کند. متدهای onPeersAvailable() برایمایز یک WifiP2pDeviceList فراهم می‌کند که می‌تواند مجددآ آن را تکرار نموده تا به دستگاهی که می‌خواهیم متصل شویم.

### 3-5-3- اتصال به جفت‌ها

هنگامی که دستگاهی که می‌خواهیم به آن متصل شویم را پس از دریافت یک لیست از جفت‌های در دسترس تعیین نمودیم، متدها connect() را به آن دستگاه فراخوانی می‌کنیم. فراخوانی این متدها نیازمند یک آبجکت WifiP2pConfig است که اطلاعات دستگاهی که می‌خواهیم به آن وصل شویم را دارا است. شما می‌توانید از موفقیت یا عدم موفقیت یک اتصال در طول WiFiP2pManager.ActionListener.

```
//obtain a peer from the WifiP2pDeviceList
WifiP2pDevice device;
WifiP2pConfig config = new WifiP2pConfig();
config.deviceAddress = device.deviceAddress;
mManager.connect(mChannel, config, new ActionListener() {

    @Override
    public void onSuccess() {
        //success logic
    }

    @Override
    public void onFailure(int reason) {
        //failure logic
    }
});
```

شکل (3-9) اطلاع از موفقیت یا عدم موفقیت در برقراری اتصال

### 3-5-4- انتقال داده

هنگامی که یک اتصال برقرار شد، می‌توان میان دستگاه‌ها از طریق سوکت‌ها داده تبادل کرد. در مرحله اول لازم است تا یک سوکت سرور<sup>1</sup> ایجاد کنیم. این سوکت منتظر یک اتصال از کلاینت روی یک پورت معین می‌ماند تا اتصال برقرار شود. پس این کار را در نخ پس زمینه<sup>2</sup> انجام می‌دهیم.

---

<sup>1</sup> Broadcast Receiver

<sup>2</sup> Server Socket

در مرحله دوم یک سوکت کلاینت<sup>1</sup> ایجاد می‌کنیم. کلاینت از آدرس IP و پورت سوکت سرور برای اتصال به دستگاه سرور استفاده می‌کند.

در مرحله سوم داده از کلاینت به سرور منتقل می‌شود. وقتی که سوکت کلاینت با موفقیت به سوکت سرور متصل شد، می‌توانید داده را از کلاینت به سرور با جریانهای بایت<sup>2</sup> منتقل کنید.

در مرحله چهارم سوکت سرور برای یک اتصال کلاینت با متده است accept() منظر می‌ماند و بلوکهایی را فراخوانی می‌کند تا یک کلاینت متصل شود. وقتی که یک اتصال اتفاق می‌افتد، دستگاه سرور می‌تواند داده را از کلاینت دریافت کند و هر عملیاتی را با داده انجام دهد نظیر ذخیره آن در یک فایل یا نمایش آن به کاربر.

کد زیر نشان می‌دهد که چگونه یک ارتباط سوکت کلاینت-سرور ایجاد کرده و تصاویر JPEG را از یک کلاینت به سرور با یک سرویس منتقل کنیم.

---

<sup>3</sup> Client Socket

<sup>3</sup> Byte Streams

```

public static class FileServerAsyncTask extends AsyncTask {

    private Context context;
    private TextView statusText;

    public FileServerAsyncTask(Context context, View statusText) {
        this.context = context;
        this.statusText = (TextView) statusText;
    }

    @Override
    protected String doInBackground(Void... params) {
        try {

            /**
             * Create a server socket and wait for client connections. This
             * call blocks until a connection is accepted from a client
             */
            ServerSocket serverSocket = new ServerSocket(8888);
            Socket client = serverSocket.accept();

            /**
             * If this code is reached, a client has connected and transferred data
             * Save the input stream from the client as a JPEG file
             */
            final File f = new File(Environment.getExternalStorageDirectory() + "/"
                + context.getPackageName() + "/wifip2pshared-" + System.currentTimeMillis()
                + ".jpg");

            File dirs = new File(f.getParent());
            if (!dirs.exists())
                dirs.mkdirs();
            f.createNewFile();
            InputStream inputStream = client.getInputStream();
            copyFile(inputStream, new FileOutputStream(f));
            serverSocket.close();
            return f.getAbsolutePath();
        } catch (IOException e) {
            Log.e(WiFiDirectActivity.TAG, e.getMessage());
            return null;
        }
    }

    /**
     * Start activity that can handle the JPEG image
     */
    @Override
    protected void onPostExecute(String result) {
        if (result != null) {
            statusText.setText("File copied - " + result);
            Intent intent = new Intent();
            intent.setAction(android.content.Intent.ACTION_VIEW);
            intent.setDataAndType(Uri.parse("file://" + result), "image/*");
            context.startActivity(intent);
        }
    }
}

```

شكل (3-10) انتقال داده بین کلاینت و سرور

در سمت کلاینت، با یک سوکت کلاینت به سوکت سرور متصل شده و داده را منتقل می‌کنیم. در اینجا ما یک فایل JPEG را روی سیستم فایل دستگاه کلاینت است انتقال می‌دهیم.

```
Context context = this.getApplicationContext();
String host;
int port;
int len;
Socket socket = new Socket();
byte buf[] = new byte[1024];
...
try {
    /**
     * Create a client socket with the host,
     * port, and timeout information.
     */
    socket.bind(null);
    socket.connect((new InetSocketAddress(host, port)), 500);

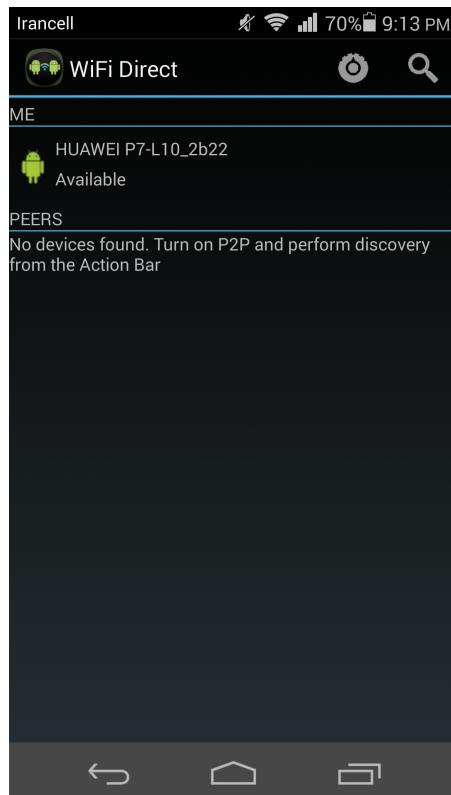
    /**
     * Create a byte stream from a JPEG file and pipe it to the output stream
     * of the socket. This data will be retrieved by the server device.
     */
    OutputStream outputStream = socket.getOutputStream();
    ContentResolver cr = context.getContentResolver();
    InputStream inputStream = null;
    inputStream = cr.openInputStream(Uri.parse("path/to/picture.jpg"));
    while ((len = inputStream.read(buf)) != -1) {
        outputStream.write(buf, 0, len);
    }
    outputStream.close();
    inputStream.close();
} catch (FileNotFoundException e) {
    //catch logic
} catch (IOException e) {
    //catch logic
}

/**
 * Clean up any open sockets when done
 * transferring or if an exception occurred.
*/
finally {
    if (socket != null) {
        if (socket.isConnected()) {
            try {
                socket.close();
            } catch (IOException e) {
                //catch logic
            }
        }
    }
}
```

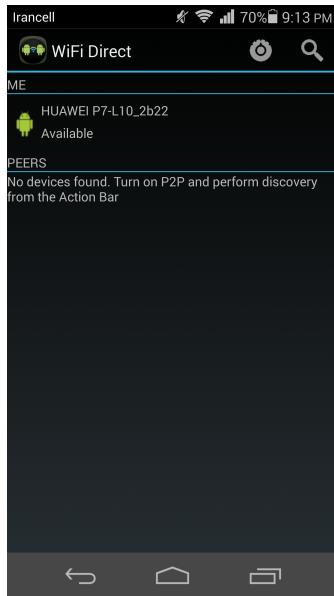
شکل (3-11) انتقال فایل JPEG

### 3-5-5- تست اپلیکیشن

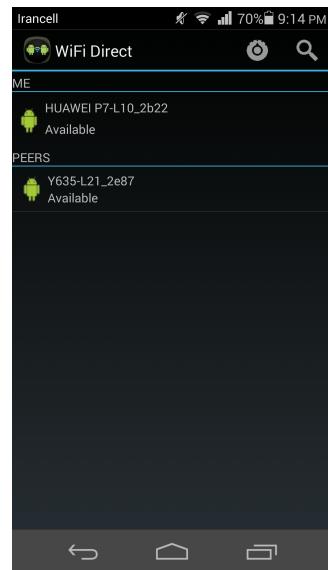
هنگامی که میخواهیم از برنامه ای که در اندروید استودیو نوشته ایم خروجی بگیریم دو راه برای تست آن وجود دارد. در صورت در دسترس نبودن یک تلفن هوشمند میتوان از یک شبیه ساز که اندروید استودیو ارائه میدهد استفاده نمود و یا میتوان مستقیماً آنرا بر روی گوشی که از طریق USB به کامپیوتر وصل شده نصب و تست نمود. هنگامی که گوشی را به کامپیوتر وصل ننمودیم لازم است تا از قسمت تنظیمات آن به رفته و تیک گزینه *i* USB debugging را بزنیم. آن‌گاه با کلیک بر روی دکمه *i* Developer Options مثلثی سبز رنگ در اندروید استودیو و یا استفاده از کلید های ترکیبی Shift + F10 اقدام به گرفتن خروجی APK نمود. در نهایت با انتخاب گوشی اپلیکیشن بر روی آن نصب شده و میتوانیم آنرا تست کنیم. در زیر نمونه ای از تست اپلیکیشن نشان داده شده است. یک دستگاه به عنوان کلاینت در خواست اتصال میدهد و پس از یافتن یک لیست از دستگاه های در دسترس و انتخاب دستگاه مورد نظر به عنوان سرور میتوان با کلیک بر روی دکمه *i* Connect به آن وصل شده و تصاویر JPEG را ارسال نماییم.



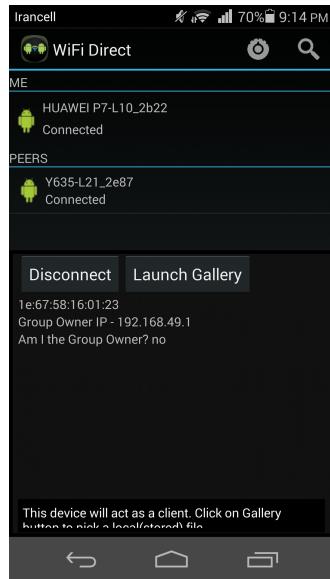
شکل (3-12) نمایی از اپلیکیشن



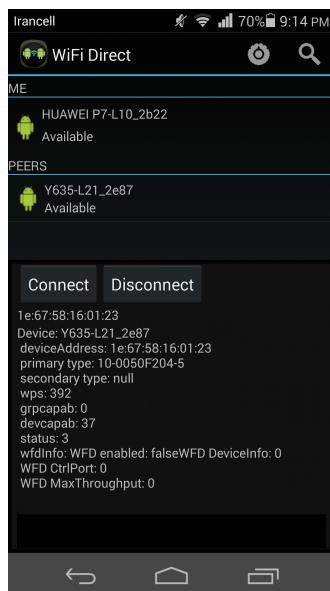
شکل (3-13) نمایی از اپلیکیشن



شکل (3-14) دریافت لیستی از دستگاه های موجود با کلیک بر روی دکمه جستجو و اتصال به آن با کلیک  
بر روی دکمه Connect پس از انتخاب نام دستگاه



شکل (3-15) اپلیکیشن در سمت کلاینت پس از اتصال به یک دستگاه



شکل (3-16) اپلیکیشن در سمت سرور پس از اتصال به یک دستگاه

## فصل چهارم

### نتیجه‌گیری و پیشنهادها

## 4-1- مقدمه

واقعیت مجازی به معنی خلق محیطی مجازی از صحنه‌هایی است به گونه‌ای که در واقعیت آنها را تجربه می‌کنیم. واقعیت مجازی از تکنولوژی‌های پیچیده‌ای برای رسیدن به این هدف بهره می‌گیرد و دارای هر دو کاربرد سرگرمی و موارد جدی‌تر نیز می‌باشد. خوشبختانه این تکنولوژی روز به روز در حال ارزان‌تر شدن است و انتظار می‌رود در آینده‌ای نه چندان دور کاربردهایی نو و ابتکاری‌تر از آن ببینیم.

## 4-2- محتوا

در این پروژه به بررسی و ساخت یک بازی واقعیت مجازی با استفاده از موتور بازی ساز یونیتی برای عینک واقعیت مجازی گوگل که با نام مقوای گوگل مشهور است پرداختیم.

### 4-2-1- جمع‌بندی

نمایشگرهای واقعیت مجازی گوناگونی تا کنون ساخته و عرضه شده است که از تکنولوژی‌های پیشرفته و پیچیده‌ای برای نمایش یک محیط مجازی شبیه سازی شده توسط کامپیوتر به کاربر بهره می‌گیرند. این نمایشگرها خود دارای پردازنده‌های قوی و سایر ملزومات می‌باشند اما رویای دستیابی به یک تجربه‌ی واقعیت مجازی توسط همگان تا پیش از ارائه نمایشگر مقوایی گوگل محقق نشده بود. در این نمایشگر شرکت گوگل به جای استفاده از سنسورها و پردازنده‌های گران‌قیمت از پردازنده و سنسورهای گوشی‌های هوشمند بهره می‌گیرد. بدین ترتیب نمایشگر واقعیت مجازی چیزی بیش از یک تکه مقوای گوگل محقق نیست که آن هم هزینه‌ی چندانی ندارد و از طرفی امروزه هر فردی دارای یک گوشی همراه هوشمند نیز می‌باشد بنابراین استفاده‌ی عموم مردم از اپلیکیشن‌های واقعیت مجازی با ایده‌ی مقوای گوگل محقق گردید.

شرکت گوگل برای اینکه عده‌ی زیادی امکان توسعه بازی‌ها و اپلیکیشن‌های واقعیت مجازی را داشته باشند یک کیت توسعه نرم افزار برای نمایشگر مقوایی خود نیز عرضه کرد که شامل API‌ها و همچنین نمونه‌هایی است که به توسعه دهنده‌گان جهت ساخت بازی یا اپلیکیشن واقعیت مجازی کمک فراوانی می‌کند.

موتورهای بازی ساز گوناگونی امکان ساخت بازی واقعیت مجازی برای این نمایشگر را برای ما فراهم نموده‌اند که در این پروژه ما به سراغ یکی از قدرتمندترین‌شان یعنی موتور بازی ساز یونیتی رفتیم. کیت توسعه نرم افزار واقعیت مجازی گوگل که برای یونیتی عرضه شده است حاوی اسکریپت‌ها و فایل‌هایی است که بازی ما را تبدیل به یک بازی واقعیت مجازی سازگار با مقوای گوگل می‌کند.

در این پروژه سعی شده است تا از تمام کاربردهایی که کیت توسعه نرم افزار به ما ارائه می‌دهد در محیط بازی خود بهره گیریم از جمله امکان تعامل با آبجکت‌های درون بازی هم با استفاده از خیره نگاه کردن هم با استفاده از دکمه‌ی آهنربایی تعییه شده بر روی نمایشگر. پس از ایجاد محیط بازی با استفاده از قالبی که تهیی نموده‌ایم امکان راه رفتن خودکار با استفاده از فشردن دکمه‌ی آهنربایی و نیز تعامل کاربر با یکسری آبجکت‌های بازی با استفاده از خیره نگاه کردن به آنها بهره گرفتیم.

## 4-2-2- نوآوری

در این پروژه در پاسخ تعامل کاربر با آبجکت‌های بازی از اجرای انیمیشن و بهره‌گیری از سیستم ذره یونیتی برای ایجاد آتش استفاده شده است.

برای کاراکتر Crawler، پس از هر بار خیره نگاه کردن کاربر به آذیکی از چند انیمیشن تعریف شده در ماشین حالت اجرا شده و دوباره به حالت اولیه خود باز می‌گردد. پس از اجرای آخرین انیمیشن نیز دوباره به سراغ انیمیشن اولاً باز می‌گردد.

برای آبجکت Cauldron، از سیستم ذره یونیتی برای ایجاد آتش بهره بردیم که مفصل‌اً در فصل دوم توضیح داده شده است و پس از نگاه کردن کاربر به آبجکت مورد نظر که یک دیگ می‌باشد آتشی از آذ خارج می‌گردد.

## 4-2-3- پیشنهادها

در تکنولوژی واقعیت مجازی گوگل برای پردازش بازی و مسیریابی حرکت سر کاربر تماماً از سنسورها و پردازشگر تلفن هوشمند استفاده می‌شود. می‌تواند به عنوان قدمی فراتر از این روش معمولاً پردازش بازی را به عهده کامپیوتر گذاشت و با گرفتن اطلاعات مربوط به سنسورهای گوشی و ارسال آذ به کامپیوتر حرکات سر کاربر را شناسایی نمود. پردازش بازی بر روی کامپیوتر این امکان را می‌دهد که چندین کاربر با داشتن نمایشگر به طور همزمان از یک بازی یا اپلیکیشن استفاده نموده و از طرفی هم سرعت پردازش در مقایسه با گوشی‌های هوشمند بسیار بیشتر است و در نتیجه می‌توان بازی‌های سنگین‌تر و با گرافیک بالاتر ایجاد نمود. ارتباط دستگاه اندرویدی و کامپیوتر می‌تواند از طریق واي-فای دایرکت یا همادواری-فای P2P صورت گیرد که به عنوان یک کلاس API هم برای سیستم عامل اندروید و هم ویندوز در دسترس قرار دارد. در فصل سوم به چگونگی ساخت یه اپلیکیشن اندرویدی که از این API استفاده می‌کند پرداخته‌ایم.

در تکنولوژی واقعیت مجازی گوگل در حال حاضر راه رفتن کاربر وجود ندارد و فقط حرکات سر کاربر شناسایی شده و در محیط می‌توان به اطراف تحت زاویه 360 درجه نگاه کرد و ما از دکمه‌ی آهنربایی برای فعال و غیر فعال کردن راه رفتن در بازی استفاده می‌کنیم. اما چه می‌شد اگر کاربر با راه رفتن در محیط واقعی راه رفتن در فضای مجازی را کنترل می‌کرد؟ یک بازی واقعیت مجازی بر روی تلفن همراه هوشمند پیاده‌سازی می‌شود تا سیستم کنترلی جدیدی را نمایش دهد. کاربر می‌تواند تجسم سه بعدی واقعیت مجازی نمایش داده شده بر روی یک نمایشگر واقعیت مجازی نظیر مقوای گوگل را کنترل کند. این امر از طریق حرکت سر، چرخیدن و راه رفتن میسر می‌شود. الگوی داده‌ی بلادرنگ از سنسورهای شتاب سنج و ژیروسکوپ درون‌گوشی هوشمند جمع‌آوری می‌شود و برای شناسایی حرکت سر کاربر برای کنترل آبجکت بازی واقعیت مجازی استفاده می‌شود. انواع مختلفی از حرکات وجود دارد از جمله راه رفتن، چرخش بدن و حرکت سر هنگامی که به چپ، راست، بالا و یا پایین نگاه می‌کند. حرکت بسته به جهت دید کاربر دارد. برای شناسایی حرکت سر می‌تواند از سنسور ژیروسکوپ استفاده نمود در حالی که برای شناسایی حرکت بدن باید از ترکیبی از سنسورهای ژیروسکوپ، شتاب سنج و معناظطیس سنج استفاده نمود. در نهایت با گرفتن این اطلاعات کاربر می‌تواند بازی واقعیت مجازی را کنترل کند.

## **فهرست منابع**

## منابع فارسی

[www.myvr.ir/virtual-reality/](http://www.myvr.ir/virtual-reality/)

## منابع غیر فارسی

[www.en.wikipedia.com/wiki/virtual\\_reality](http://www.en.wikipedia.com/wiki/virtual_reality)

[www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html](http://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html)

[www.developers.google.com/vr/cardboard/overview](http://www.developers.google.com/vr/cardboard/overview)

[www.dzone.com/articles/develop-a-google-cardboard-virtual-reality-applica](http://www.dzone.com/articles/develop-a-google-cardboard-virtual-reality-applica)

[www.makezine.com/projects/getting-started-with-virtual-reality-an-introduction-to-unity](http://www.makezine.com/projects/getting-started-with-virtual-reality-an-introduction-to-unity)

[www.makezine.com/projects/getting-started-with-virtual-reality-building-for-google-cardboard](http://www.makezine.com/projects/getting-started-with-virtual-reality-building-for-google-cardboard)

[www.makezine.com/projects/getting-started-virtual-reality-add-interactivity-google-cardboard](http://www.makezine.com/projects/getting-started-virtual-reality-add-interactivity-google-cardboard)

[www.raywenderlich.com/116805/make-vr-game-unity-google-cardboard](http://www.raywenderlich.com/116805/make-vr-game-unity-google-cardboard)

[www.raywenderlich.com/113049/introduction-unity-particle-systems](http://www.raywenderlich.com/113049/introduction-unity-particle-systems)

[www.assetstore.unity3d.com/en/#!/content/7004](http://www.assetstore.unity3d.com/en/#!/content/7004)

[www.assetstore.unity3d.com/en/#!/content/36269](http://www.assetstore.unity3d.com/en/#!/content/36269)

[www.assetstore.unity3d.com/en/#!/content/16674](http://www.assetstore.unity3d.com/en/#!/content/16674)

[www.developer.android.com/guide/topics/connectivity/wifip2p.html#creating-br](http://www.developer.android.com/guide/topics/connectivity/wifip2p.html#creating-br)