Проект по Мобилни приложения за Андроид

За реализацията на проекта ми по андроид съм използвал framework за създаване на игри, наречен libGDX, който предоставя класове и методи които асистират в създаването на игри.

Целта на играта е да бъдат хващани капки, които падат от върха на екрана, с кофа, която може да бъде движена чрез докосване на екрана на мобилното устройство, като тя ще следва позицията на пръста на играча.

Класовете на проекта се намират в core/src/com/mygdx/game

При стартирането на приложението се създава нова инстанция на класа GameMainClass. Този клас удължава(extends) класа Game, който казва на приложението, че това ще е игра. Веднага след като бъде създаден този клас, се създават 3 property-та , SpriteBatch batch, BitmapFont font и Music rain. SpriteBatch е класът който рисува елементите на играта върху екрана и е направен за да взема няколко текстури и да ги рисува наведнъж, като те трябва да бъдат въведени между методите begin() и end() на SpriteBatch обекта. Следващият създаден клас е BitmapFont, който предоставя фонт с който SpriteBatch да може да рисува букви и цифри. Третото property е класът Music, който взема mp3 файл от assets папката на проекта и предоставя методи за неговото ползване.

След като биват създадени променливите, в класът create() се инициализират и се създава обектът game който се подава на методите които изискват неговите property-та. Променливата rain получава като стойност rain.mp3 файлът от папката assests, като той ще бъде музиката която ще се чува в приложението. След това, font получава своята стойност и размер с метода setScale(x,y) като двете променливи определят размера му. След като текстът за приложението бива създаден, rain.setLooping(true) казва че стойността на обекта ще се повтаря като приключи, което гарантира, че винаги ще се чува валящият дъжд. След като е казано че rain ще се повтаря се включва метода play() на класа Music, който пуска стойността. След като всички тези неща са били изпълнени се включва следващият метод, който създава класът MainMenuScreen и го слага като екран за играта.

Когато всичко в create() метода бъде изпълнено се включва render() метода, който започва да рисува. В тялото на метода в момента има само едно нещо и то е super.render() който казва на GameMainClass да започне да рисува.

Една от големите промени които libGDX прави е, че премахва activities и ги замества със Screen класовете, в render() методите на които с помощта на SpriteBatch класа се рисува по екрана, като бъдат подавани координатите на които да бъдат нарисувани обектите.

След като бива сложен MainMenuScreen като главен екран на приложението, се създава негов обект чрез конструктора му, който слага GameMainClass като game и инициализира prоperty-тата на класа, които са MainGameClass, който му е нужен заради неговото property: batch,което е обект на класа за рисуване SpriteBatch, и camera който е обект на класа OrthographicCamera, който е класа който предоставя камерата за играта, като тя бива настроена след създаването си с метода setOrtho(), като той й казва че екрана към който ще гледа ще е 800 на 480 пиксела. След като бива създаден класът MainMenuScreen, се включва render() метода му. Първото нещо което прави метода е да изчисти екрана и да го направо тъмно син, след това се ъпдейтва камерата и се казва на batch, къде да рисува като се синхронизира с позицията на камерата. След като бива уведомен batch, къде да рисува се вика неговият begin() метод след който бива поздравен играча с изписване изрисуване на текст на екрана и му биват обяснени правилата и бива уведомен как да започне играта: чрез докосване на екрана.

libGDX може да засича всички видове докосвания на екрана, като те биват наричани TouchEvents и с помощта на методите статичния клас Gdx.input те могат да бъдат използвани. Тук той бива извикан в if като се казва, че ако в момента има какъвто и да било активен TouchEvent да бъде създаден и зададен като главен екран на приложението класа GameScreen, който е екрана на който е играта, като за параметър той иска game, който е обектът на MainGameClass, който е класът в който се намира SpriteBatch обекта за рисуване и веднага след това се активира методът dispose() на MainMenuScreen, който го унищожава.

След като бива поставен като главен екран на приложението се създава класа GameScreen, в който се съдържа логиката на играта. Класът има property за game, класът със SpriteBatch обекта, също така и за двете текстури за кофата и капката, едно за звукът който ще се чува когато капката се докосне до кофата, една камера, един квадрат който ще представлява кофата, един array от квадрати който ще са капките, един long който ще определя колко често падат капките, още един за това колко са събрани, който е public и static за да може да бъде използван и от екрана на който е класацията, и един последен integer който показва колко живота има играча. Когато обектът на класа GameScreen бива създаден на property-тата му биват предадени стойности, като текстурата на кофата и капката, взети от assets папката, след което се определят началните x и y на квадрата представящ кофата, също и неговите височина и ширина.

Методът SpawnRaindrop създава първата капка на играта, както и всяка след нея, като прави нов квадрат който да представя капката като му дава произволна стойност x което ще я постави всяка капка на различно място, и стойност y по-висока от тази на екрана за да изглежда сякаш капката пада от високо, а не се spawn-ва точно извън пространството което играчът вижда. След като бива създадена капката, тя бива добавена в array raindrops и се придава число което да казва кога да се появи в зависимост от последната появила се капка.

След като бива създаден класа се включва и неговият render() метод, който отново изчиства екрана и му придава тъмно син цвят и казва на batch къде да рисува.

Понеже в libGDX има точно определи методи за всички действия, се налага някои от тях да бъдат слагани в неуместни методи, където трябва ,препоръчително, да бъдат сложени за да работят правилно, като например gameplay-loop-а трябва да бъде сложен в render() методът който рисува на екрана.

След като бива изчистен и оцветен екрана, се ъпдейтва камерата на играта и започва метода begin() на batch, който изписва на екрана колко капки е уловил играча, колко живота има и изрисува кофата на координатите които са й били зададени, и със for започват да се изрисуват всички квадрати за капките и да им биват прилагани текстурите от waterdrop полето на GameScreen обекта . След като биват нарисувани тези елементи на екрана, batch проверява колко живота има играча и когато те стигнат 0, изписва на екрана, че играта е приключила, и ако е, му изписва Game Over, и инструкции как да стигне до класацията и да рестартира играта, след което стига до метода end() на SpriteBatch който приключва рисуването и започва логиката на играта.

След като провери дали играча има животи, играта проверява дали има активен TouchEvent с isTouched и ако има създава вектор, който ще съдържа координатите на позицията на пръсна на играча, след което камерата му позволява да променя координатите си когато играча мести пръста си и после задава x координата на х на кофата, което я кара да се мести заедно с пръста на играча, но само на ляво и на дясно.

Когато вече е възможно за кофата да се мести се създават два integer-а, единият от който съдържа начална скорост с която падат капките, а другият съдържа времето което трябва да измине за да падне нова капка. След създаването на тези двете се създава шаблона Итератор за raindrops и се казва че когато изминалото време от последната паднала капка надвиши времето през което падат капките, да се създаде нова капка в array-а. Докато в raindrops има капки, те ще продължат да падат, което на теория трябва да е завинаги. След като биват нарисувани капките над върха на екрана, от всяка от тях бива изваждана скоростта им на падане от y-ка им, което ги кара да се движат към дъното на екрана. След това се определя след колко хванати капки колко по бързо да падат те и колко по-често. Под тях се казва какво да стане когато една капка не бъде уловена от кофата, в този случай да бъде премахнат живот и да се включи вибратора на телефона за 200 милисекунди. Ако квадрата представящ капката и квадрата представящ кофата се докоснат ще се увеличи броя на уловените капки се пуска звукът от property-то raindrop\_sound.

Ако животите на играча паднат под 1, кофата и капките ще спрат да се движат и на екрана ще се покаже Game Over след което ще се провери за следващият включил се TouchEvent с justTouched() метода, с други думи, ако в момента има включен TouchEven, приложението ще изчака той да се изключи и когато се включи следващият методът ще върне true, тоест, когато играча вдигне пръста си от екрана и го докосне пак, ще се включи кода в if, който създава и поставя като главен екран HighScoreScreen, унищожава GameScreen и обектите в него и приравнява dropsGathered на 0 за да е готова за рестартиране играта.

Класът HighScoreScreen има няколко property-та, едното е array в който се съдържат точките на играчите, второто са точките които са събрани в предишната игра, третото е отново камерата и четвъртото е Preference, което е клас предоставен от libGDX, който може да бъде използван за запазване на малко количество данни, като настройките използвани от играчите, статистиките на герои или както в този случай точките в класацията.

Методът savePrefs() запазва трите най-високи резултата от array-я в pref обекта, а loadPrefs() ги зарежда и поставя отново в него когато се стигне до класацията.

Методът addScore използва спечелените точки в последната игра и ги слага в класацията в зависимост от това колко повече или по-малко са от тези които са вече в нея. Също така и премахва точките които не са по-високи от тези вече в array-я за да го поддържа само с 3 стойности в случай че се появят нежелани bug-ове относно големината на array-я и тази на таблицата с класациите.

След като бъде създаден обекта от класа HighScoreScreen се включва методът render() който, както и при останалите изчиства екрана, осветява го в синьо и казва на batch къде да рисува. Когато batch получи своите инструкции къде да рисува се извиква begin() метода му, който изписва най-високите резултати от array-я HighScores и инструкции за това как да се рестартира играта. После се проверява за започването на следващият TouchEvent, при стартирането на който се рестартира играта.

Когато се напусне играта се извиква dispose() метода на game който е обекта на GameMainClass и го унищожава, спирайки играта.