## A képen szöveg, képernyőkép, szoftver, Betűtípus látható Automatikusan generált leírásA képen szöveg, diagram, Párhuzamos, Tervrajz látható Automatikusan generált leírás-kód bemutatása

3. ábra Osztály diagram

Az itt látott kódrészlet a Specific osztály 2 függvényé az egyik a distnace a másik a dxdy nevet viseli. A distance függvény mint már említettem az objektumtól való másik objektum légvonal béli távolságot hivatott kiszámolni mind ezt az által, hogy a MyGLRendererben található midleCoordinate függvényt használva, ami visszatér a 2 objektum x és y koordinátán való elhelyezkedésével és ezek alapján egy egyszerű távolság számító függvénnyel kiszámítja ezen 2 objektum távolságát. Ezen függvény természetesen az öröklődés miatt működik mindegyik játékbéli objektumon. A distance metódus szolgált a fejlesztés elején az ütközések kezelésére viszont ezt idővelleváltotta BoundingBox illetve a BoundingCircle osztályok melyek precízebb és hibamentesebb megoldást nyújtanak. A másik függvény, ami még a képen látható az nem más, mint a dxdy függvény, ami főleg az ellenfeles objektumok miatt lett beépítve. Ugyan is ezen függvény főbb célja az, hogy a 2 dimenziós térben, ahol a játék zajlik ezen függvényen keresztül képesek legyünk megmondani, hogy a többi objektum milyen irányba A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leíráshelyezkednek el az adott objektumhoz képest. Kérdezhetnénk, hogy „Miért volt erre szükség?” amire csak annyi választ lehetne adni, hogy ezen a függvényen keresztül képesek vagyunk megmondani az ellenfeleknek, hogy merre találják a játékost, ha arra lenne szükség vagy esetleg a játékosnak kéne valami külső eseményre reagálni, ami megint csak ez alapján lenne megoldható. Ami még fontos függvény lehet ebben az osztályban az a getHeight függvény lehet, amit azon indokból írtam meg hogy ha a programban valami eltolásra lenne szükség, ami a háttér és a benne található négyzetese elemek által teremtett négyzetrácsos rendszerben való eltolás akkor se teremtsen A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásproblémát, ha az adott négyzetes elemek mérete esetlegesen a fejlesztés folyamán elkezdenek változni. A függvényben látható allCoordinates metódus segít ezen megoldás eléréséhaz. Mert maga az előbb említett metódus annyit csinál, hogy a kapott objektumon végrehajtja a programban felirt átméreteződést ami ugye kihatással lehet az objektum méreteire. Az ebből a függvényből visszatérő koordinátákból kivontam azokat, amik a négyzet tetejének és aljának felelt meg majd ennek vettem abszolút értékét arra az esetre, ha ezen objektum valamilyen forgatási művelet folyamán fejjel lefele helyezkedne. A következő képen már a SpriteSheets osztály konstruktora található, ami 4 bemeneti paraméterrel rendelkezik amik nem lennének mást hivatott reprezentálni mint a spritesheet-hez való elérési utat az ezen található texturák szélességét, magasságát és az ehhez tartozó FPS számot ami egyszerűen csak annyit jelentene hogy milyen sebességgel történjenek ezen textúrák közötti váltások a játék futása során. Ezen adatok letárolását követően történik az adott spritesheet beolvasása a BitmapFactory és Bitmap osztályok segítségével. Amikkel a folyamat úgy néz ki pontosan, hogy a bitmap nevű változóban letárolódik az egész spritesheet majd ezen bitmapből a magasságnak és szélességnek megfelelő négyzeteket vágunk ki, amiket azonnal be is töltjük az OpenGLES-be majd ezt követően az OpenGLES által hozzá generált id-t letároljuk soronként a spriteSheetArray-bea fent látható módon. Ezen módszert eredetileg helyettesíteni tudtam azzal, hogy az adott textúrákat kivagdostam külön png fájlokba majd ezeket betöltve az Android Studio-ba egyesével használtam A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírásőket. De az egyre növekedő textura mennyiségek és a konzulensem tanácsára elkészítettem ezen osztályt, amely megszüntette a további hosszadalmas és aprólékos megoldás szükségletét. A lejjeb(ábraszám) látható képen továbbra is a Spritesheet osztályt láthatjuk viszont ezek már a hozzá tartozó függvényeket, amik az adott spritesheet-ből kinyert képeket kezeli az előbb említett időleges textura váltogatással. Mint látható a képen az a nevű változóba töltődik be az előbb említett id-k ból a kívánt textura méghozzá a szerint hogy a paraméter listában kapott irany nevű változó 0,1,2,3 értékkel rendelkezik ha ezen változó olyan értékkel érkezik ami nem felel meg a betöltött textúrák mennyiségének akkor a függvény egyszerűen ignorálja az id lecserélését és az inicializált értékkel halad tovább ami nem lehet üres hiszen az osztály konstruktoraiból egyik sem rendelkezik olyan verzióval amely ezt megengedné hisz nem megfelelő sourceId-val indított inicializáció futásközben hibát dob. A metódusban láthatjuk még továbbá az előbb említett időleges váltakozást, ami úgy valósul meg hogy az adott texturán eltöltött időt számolja a függvény és ha ez meghaladj a konstruktorban megadott időt akkor a counter érték növekszik ami által a következő textura történik betöltésre. Lejjebb még látható az ezen metódus paraméter lista nélküli változatát, ami csak a legelső textura elemet mutatja. Ezen függvényre azért volt szükség mert a tesztelés folyamán nem minden objektum kapott irányra és mozgásra megfelelő textúrát igy ezen osztály ilyenkor egy általános képpel töltődik be aminek ugye nincs szüksége az előbb említett funkciókra. A követkeA képen szöveg, képernyőkép látható

Automatikusan generált leírás