**Софийски университет „*Св. Климент Охридски*“**

Факултет по математика и информатика

Вградени и автономни системи

Курсов проект

My Accelerometer Game

|  |  |
| --- | --- |
| Представил:  Анастасия Антонова  Информатика, IVти курс  45171 | Ръководител:  Проф.Васил Цунижев |

Съдържание

[Увод 3](#_Toc30880187)

[Хардуерни изисквания 3](#_Toc30880188)

[Софтуерен анализ 4](#_Toc30880189)

[Потребителски интерфейс 7](#_Toc30880190)

[Начален екран 8](#_Toc30880191)

[Ниво 1 9](#_Toc30880192)

[Ниво 2 10](#_Toc30880193)

[Ниво 3 11](#_Toc30880194)

[Ниво 4 12](#_Toc30880195)

[Предимства на играта 13](#_Toc30880196)

[Бъдещи разработки 13](#_Toc30880197)

[Препратки 14](#_Toc30880198)

# Увод

My Accelerometer Game е игра, която използва възможностите на вградения в едно Андроид устройство[[1]](#endnote-1) акселерометър[[2]](#endnote-2). Самата игра представлява движещо се човече в лабиринт, което трябва да достигне до зададена цел, за да спечели. По пътя му може да има препятствия под формата на стени, врагове и др.

Целта на проекта е да се демонстрира как работят вградените сензори и тяхното последващо използване.

Проектът е базиран на следните платформи и технологии:

* *Android OS* – много адаптивна и общодостъпна платформа с отворен код
* *Акселерометър* - уред(сензор), чрез който се измерва ускорението на движещи се обекти. Чрез него се измерва нарастването на механичната величина. Тя е инструментален метод за измерване на ускоренията. С негова помощ се измерва наклона, под който се намира Андроид устройството.
* *JBox2D[[3]](#endnote-3) -* библиотека за моделиране на физическите обекти (тела) в 2D пространството. Представлява порт на известния C++ софтуер[[4]](#endnote-4) за симулация на твърди тела. С нейна помощ обектите се движат и могат да бъдат уловени събития на сблъсък.

# Хардуерни изисквания

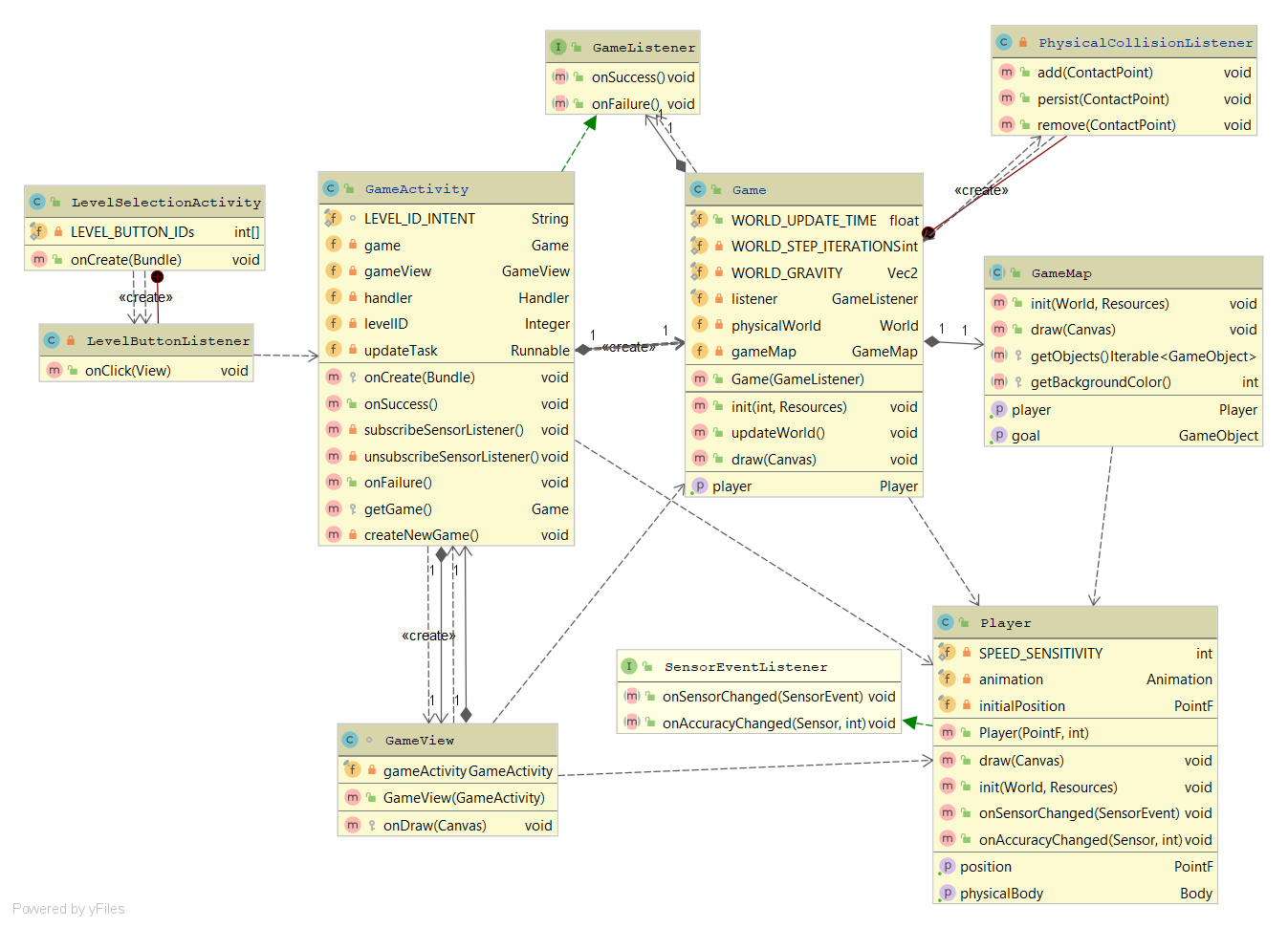
Играта за да може да бъде играна, трябва да бъде инсталирана на мобилно устройство със следните изисквания:

* операционна система *Android* , версия поне 5.0 (API 21[[5]](#endnote-5)) ,
* вграден екран
* вграден акселерометър

Повечето съвременни преносими Андроид устройства са снабдени с Акселерометър и имат наличен достатъчно голям екран, за да може играта да се играе спокойно и без никакви проблеми.

# Софтуерен анализ

Софтуера за Андроид устройства се пише с помощта на Android Software Development Kit (SDK)[[6]](#endnote-6), който е базиран на Java[[7]](#endnote-7) технологията. Java сама по себе си е език, който ни дава големи възможности за разработка, благодарение на многото си безплатни библиотеки и функционалности. Освен това приложенията написани на този програмен език за лесни за реализация и последваща поддръжка.

Нашето приложение има няколко основни класа и много унаследяващи ги такива.

Фигура 1UML диаграма на основните компоненти на приложението

*Фигура -UML диаграма на основните компоненти на приложението*

LevelSelectionActivity е първоначалният(main) изглед, който се визуализира при стартиране на приложението. В него са наредени четири бутона за всяко от четирите нива. Към всеки от бутоните е закачен LevelButtonListener, който се активира при натискане на бутон. При активация Listener-a отчита идентификатора на натиснатия бутон и стартира GameActivity.

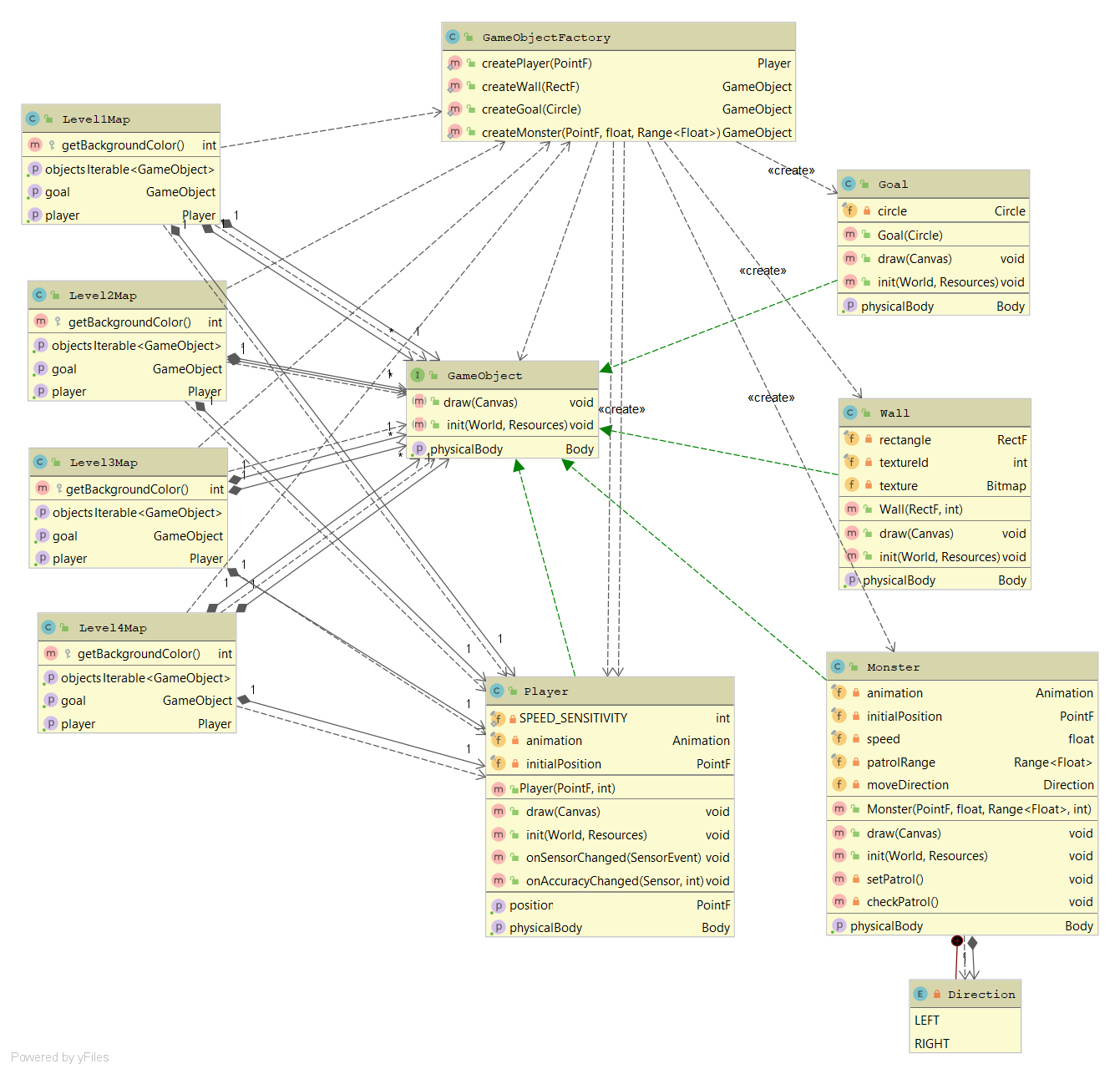
GameActivity отговаря за основния игрови изглед, в който се визуализира играта. Създава основния обект Game, който управлява самата игра. В GameActivity e дефиниран и Handler[[8]](#endnote-8) (нишка която изпълнява постъпилите заявки). Той се грижи за постоянното опресняване на играта (60 пъти в секунда).

Класът Game се грижи да осигури физическия свят с помощта на библиотеката JBox2D. Отговорен е за създаването на игровата карта, нейното инициализиране и визуализиране, включително и на всички останали игрови обекти. При извикването на метода updateWorld се опреснява физическият свят. Освен това с обект от типа PhysicalColisionListener се следи за наличието на сблъсъци между обектите във физическия свят. Ако се установи наличие на сблъсък се проверява дали той е между играча и целта, в този случай се установява наличие на успех , визуализира се съобщение за успех, GameActivity приключва и на яве изскача отново LevelSelectionActivity. Ако обаче е установен сблъсък между играча и друг обект, който не е целта, в този случай играта приключва и Game обектът се пресъздава.

Класът GameMap представя игровата карта, която съдържа самите игрови обекти. Той се грижи за инициализацията на обектите във физическия свят както и за тяхното последващо изрисуване. Самият клас си има собствен фон, който се рисува преди всички останали обекти и има референция към обекта играч и цел, за да може по късно тази референция да се използва от класа Game.

Player това е самият играч, това е и класът, който следи за промени в отчитането на акселерометъра, с помощта на EventListener[[9]](#endnote-9). Класът Player имплементира SensorEventListener[[10]](#endnote-10) интерфейса и с помощта на метода onSensorChanged бива уведомен за възникването на всяка промяна в сензорните резултати. По този начин се подава вектор с , който се прилага върху физическото тяло на обекта играч и с негова помощ той бива преместен. Самият играч има в себе си зададена начална позиция, а променливата SPEED\_SENSITIVITY променя дължината на вектора з движение с цел адекватно движение при адекватен наклон.

Класът GameView представлява самият изглед. Бива опресняван от Handler-a в GameActivity обектът. Предава обект Canvas към Game обектът, косвено към игровата карта и обектите в нея с цел тяхното изчертаване. Другата му задача е да променя позицията си спрямо позицията на играча с цел той винаги да се намира в центъра на екрана.



Фигура 2 UML диаграма на игровите обекти

*Фигура - UML диаграма на игровите обекти*

В основата на тази диаграма седи абстракцията GameObject. Той има физическо тяло инициализирано във физически свят и може да бъде рисувано с метода draw. Съществуват четири имплементации: Player, Goal, Monster, Wall. Всяка от тях има свои собствени спецификации както следва:

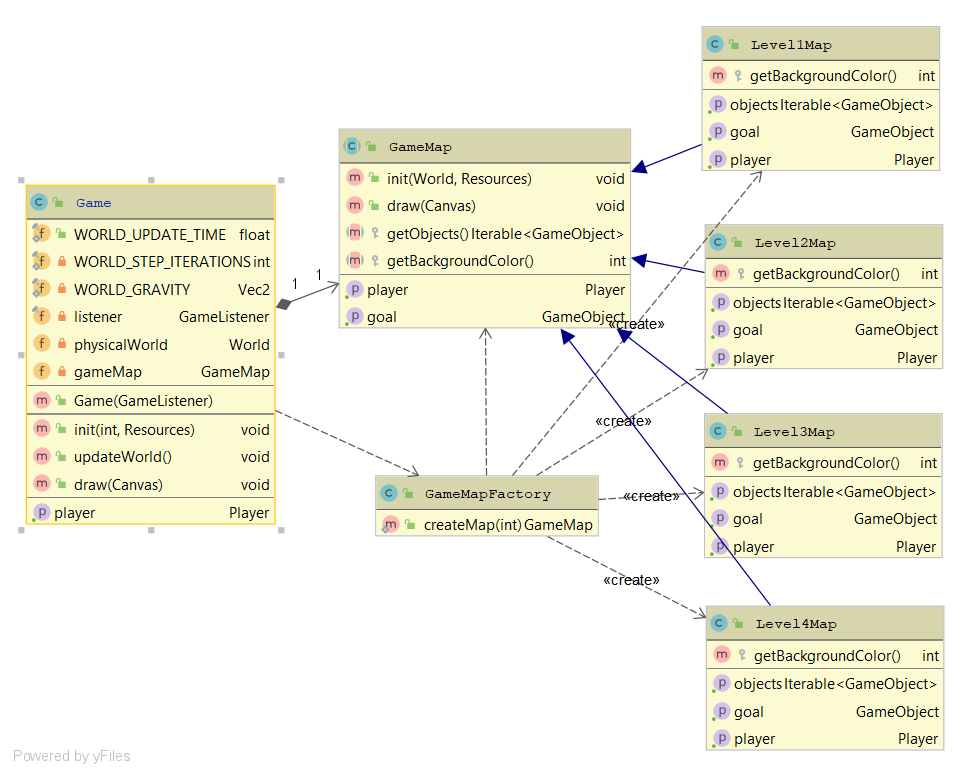
Играчът има начална стартова позиция, физическо тяло, анимация и константата SPEED\_SENSITIVITY която отговаря за скоростта с която се движи играчът при определен наклон на мобилното устройство.

Целта е представена с кръг и физическо тяло за регистриране на сблъсъци.

Стена – това е правоъгълник с текстура, в случая картинка, и физическо тяло.

Чудовище – то също се визуализира като анимация, има физическо тяло, начална позиция и обхват на движението. В конкретния случай чудовищата се движат по х оста със зададена начална и крайна точка и посока на движение. В момента когато стигне някоя от крайните точки посоката на движение се променя, проверка за достигане до крайна точка се прави при всяко опресняване на обекта.

Всички тези обекти биват създавани единствено от GameObjectFactory, имплементиращ известният factory pattern[[11]](#endnote-11), с цел повишаване нивото на абстракция. Различните карти създават обекти използвайки фабриката и държат техните референции в колекции.



Фигура 3UML диаграма на игровите карти

*Фигура - UML диаграма на игровите карти*

В основата седи абстрактния клас GameMap, той подсигурява инициализацията на всички игрови обекти и последващите им опреснявания. Има свой собствен фон, който се рисува преди всички останали обекти, и държи референция към Играча и Целта, които в последствие се използват от класа Game по време на колизия. Самите имплементации на картите от първата до последната дефинират различни обекти с различни координатни конфигурации. Те биват създавани от GameMapFactory фабриката, за да се повиши нивото на абстракция. Разграничаването на това коя карта да бъде създадена се прави спрямо идентификатора на натиснатият бутон в началното Активити. Game използва GameMapFactory при инициализацията си за да създаде GameMap.

# Потребителски интерфейс

## Начален екран



Началния екран на играта представлява навигационно меню, от което може да се избират различните игрови нива.

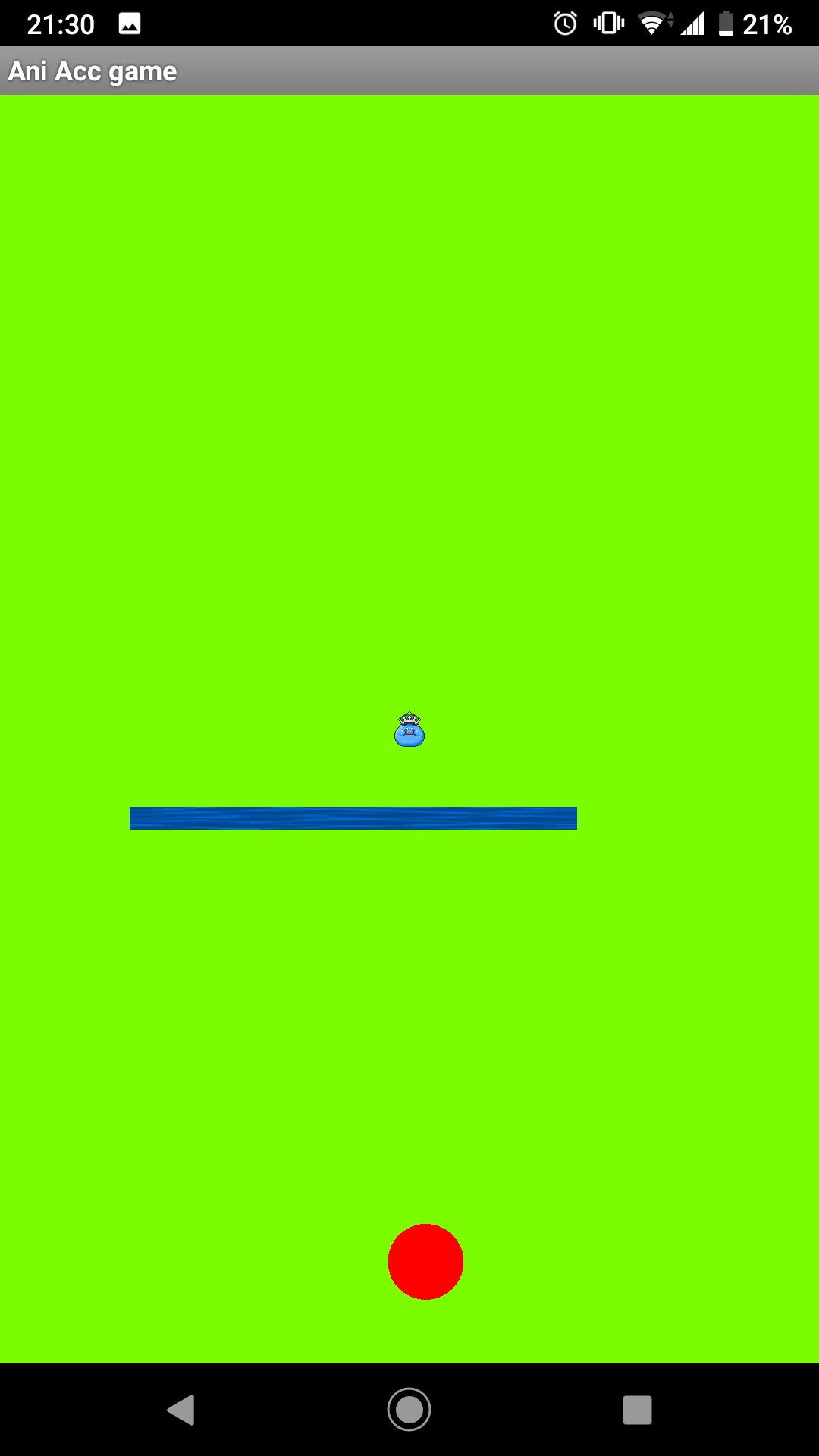
При натискане на който и да е бутон се отваря съответното ниво за игра.

Когато играч приключи с дадено ниво, приложението го препраща обратно в това меню.

Фигура 4 Снимка на началния екран - меню

*Фигура - Снимка на началния екран - меню*

## Ниво 1

Ниво 1 е първото и най-лесно ниво в играта.

Можем да видим синьо геройче с коронка – това е главният играч.

Неговата цел е да стигне до своята палатка – червено кръгче, наподобяващо купол.

Играчът се намира на зелено поле, по средата на което тече река – т.нар. препятствие.

За да се движи играчът, мобилното устройство трябва да бъде наклонено в посоката на желаното движение.

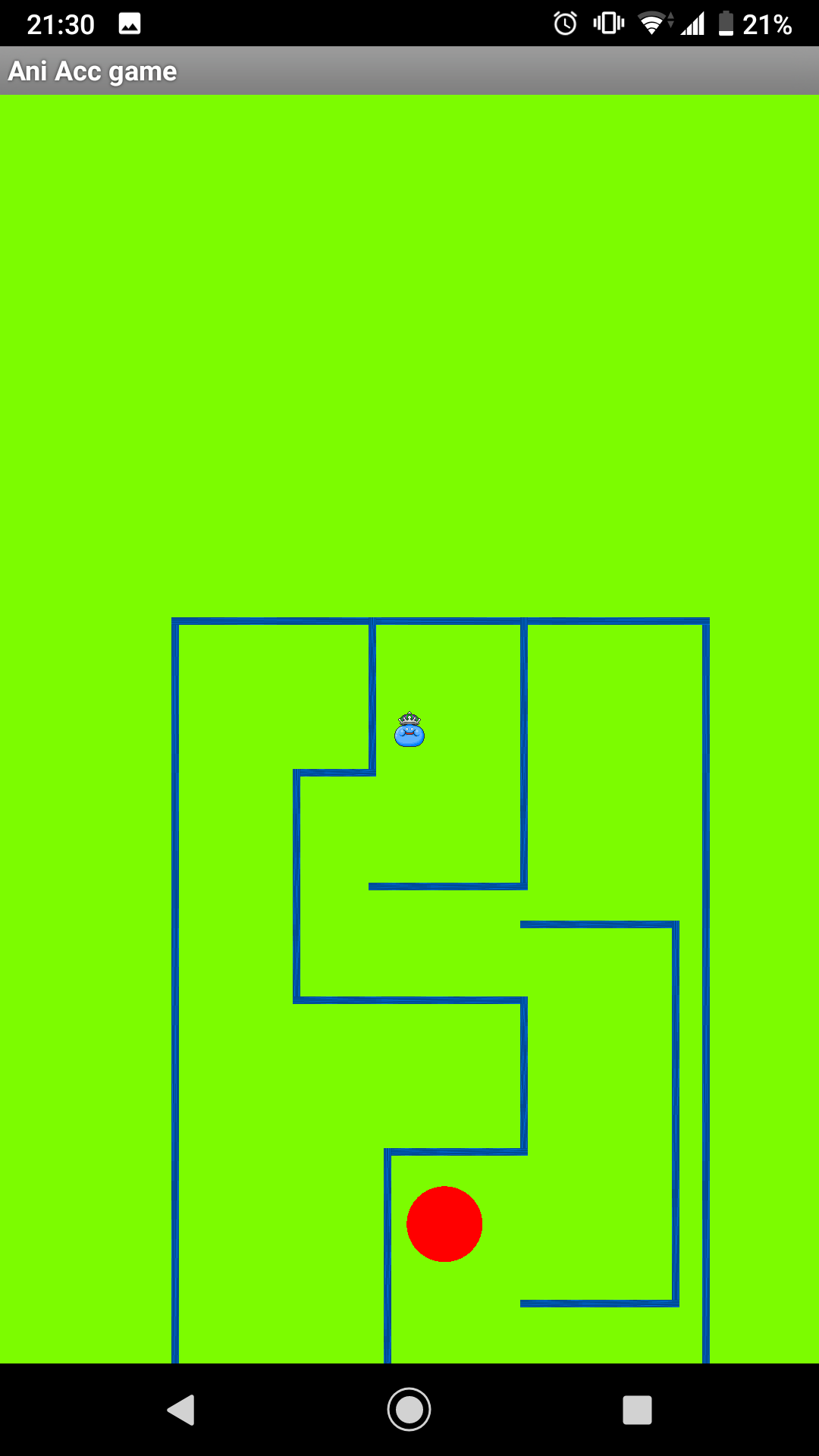
За да стигне до целта си, играчът трябва да избегне/заобиколи препятствието.

Ако при някаква случайна злополука играчът падне в реката, той бива върнат от играта обратно на начална позиция – нивото се рестартира.

Фигура 5- Снимка на Ниво 1

*Фигура - Снимка на Ниво 1*

## Ниво 2

В Ниво 2 идеята наподобява тази в Ниво 1.

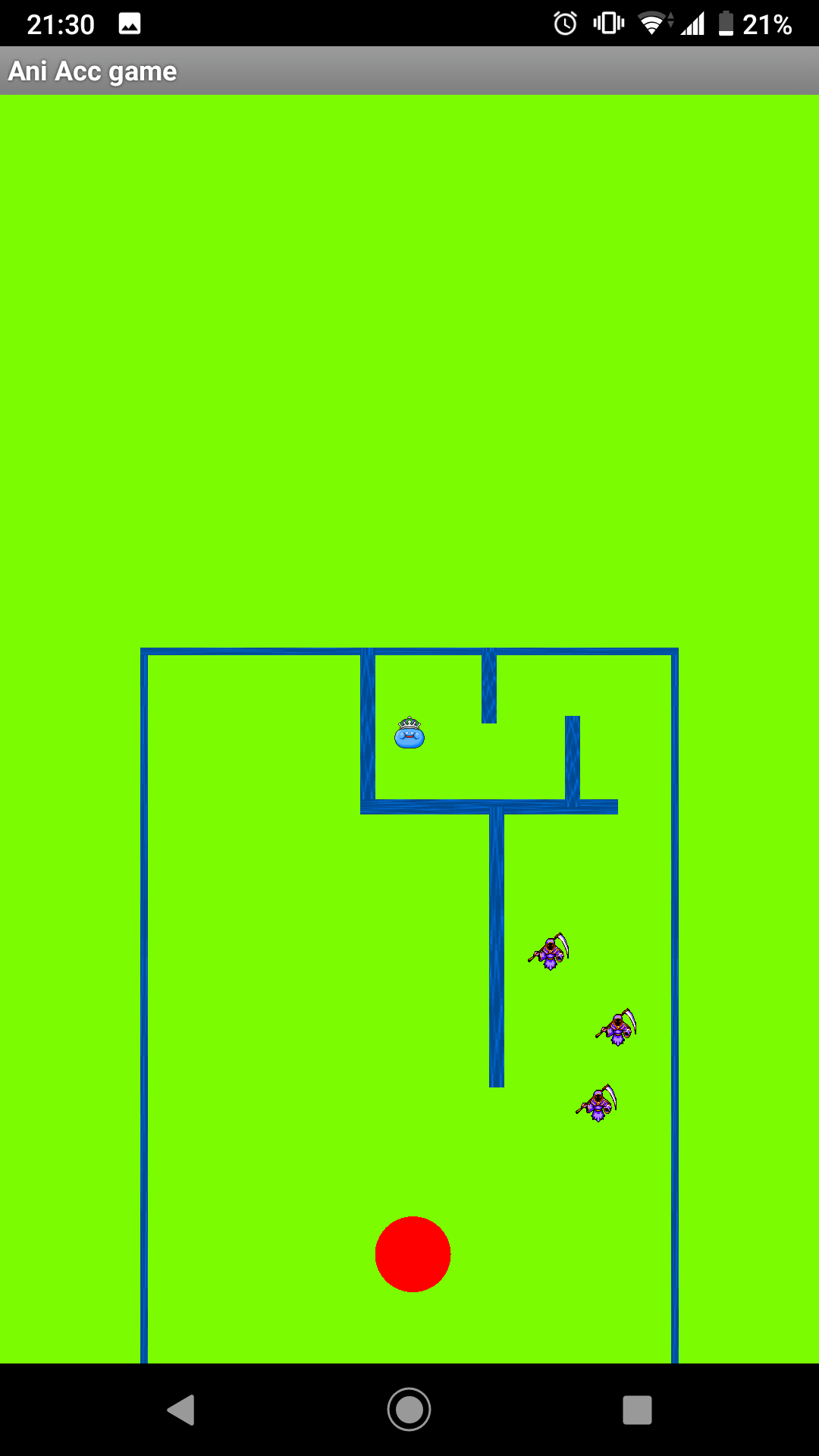
Новото тук е, че на тази земя реките са повече.

Играчът трябва внимателно да подбере пътят си , за да оцелее и да се прибере невредим в палатката си.

Фигура 6 Снимка на Ниво 2

*Фигура - Снимка на Ниво 2*

## Ниво 3



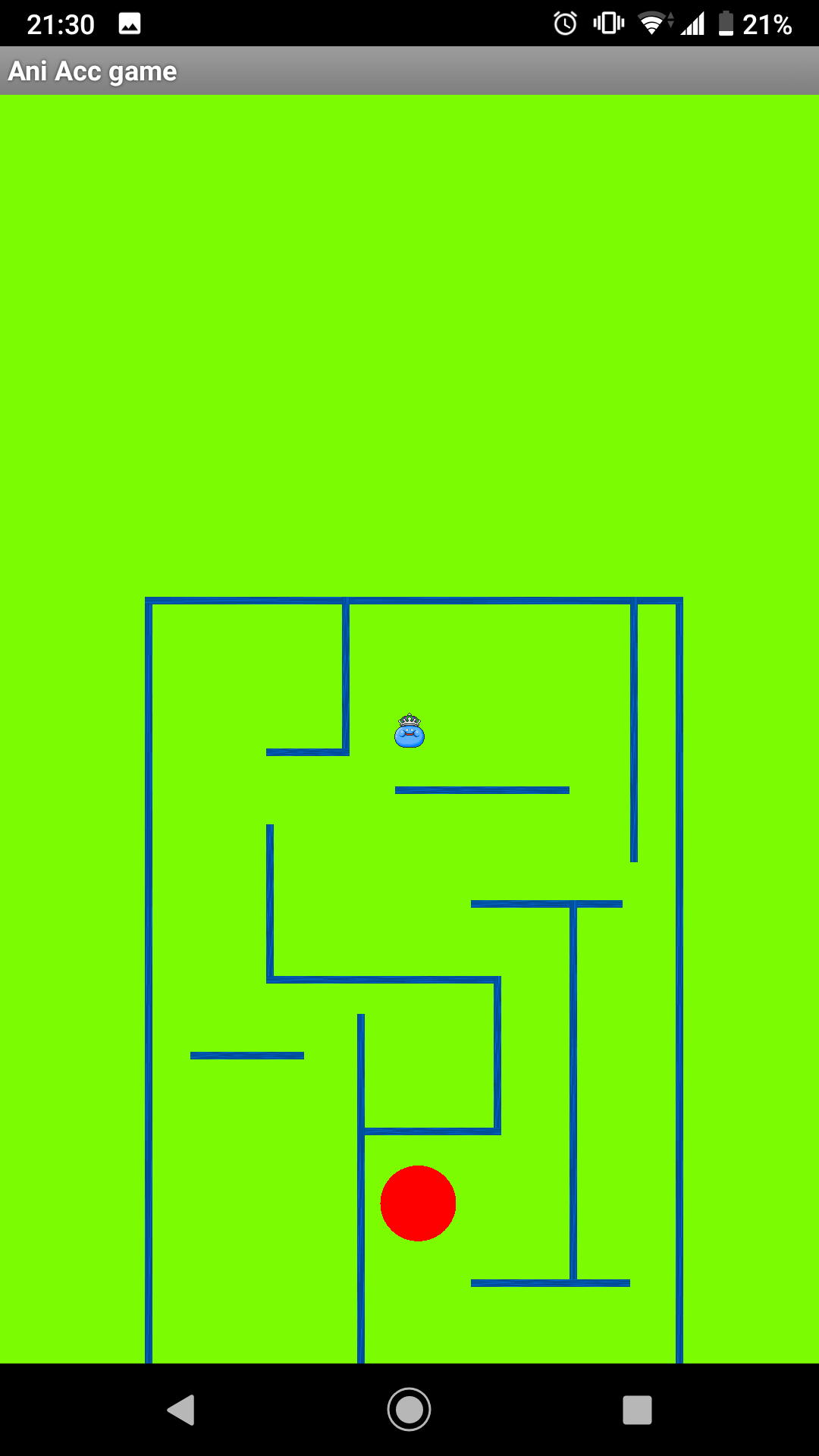
Трето ниво става малко по- интересно – тук вече са налични и зли демони, които се движат и пречат на геройчето да стигне до целта си.

За да се премине през нивото се изисква прецизност и търпение.

Фигура 7 Снимка на Ниво 3

*Фигура - Снимка на Ниво 3*

## Ниво 4

Ниво 4 е последното и най-трудно (за сега) ниво.

Рекичките тук са доста по-гъсти, следователно и проходите са доста по-тесни.

За да се премине нивото са необходими доста прецизни движения.

Фигура 8 Снимка на Ниво 4

# Предимства на играта

Играта притежава четири нива, всяко от които с различно ниво на трудност. Като започнем от първото което е най-лесно и ни дава обща представа за това как да играем, минем през второто където има по сложен лабиринт до достигане на целта. Третото ниво е там където идват на помощ така наречените чудовища и играча трябва да внимава да не се сблъска с някое от тях, а последното е отново само лабиринт с по-сложна структура.

Друго много важно нещо за играта е, че играчът се намира винаги в центъра на екрана, по този начин се добива представата, че се движат стените, а не играча но така и се улеснява възприятието за движение на играча с помощта на данните идващи от акселерометъра.

И накрая в играта има не само статични обекти препятствия, но и подвижни такива, които затрудняват допълнително нивата на играта.

# Бъдещи разработки

Впоследствие играта може да бъде разширена. Например лесно можем да добавим нови обекти от тип чудовища с други възможности за движение, можем да добавим още нива с по-висока трудност. Има възможност да дадем допълнителен контрол на играча за движение в ляво или дясно с помощта на натискане на екрана а в един от двата края. Като бонус може да се имплементира и възможност за игра в пейзажен режим на екрана.

# Препратки

1. "Android device." <https://www.android.com/intl/en_uk/>. Accessed 25 Jan. 2020. [↑](#endnote-ref-1)
2. "Accelerometer - Wikipedia." <https://en.wikipedia.org/wiki/Accelerometer>. Accessed 25 Jan. 2020. [↑](#endnote-ref-2)
3. "JBox2D: A Java Physics Engine." <http://www.jbox2d.org/>. Accessed 25 Jan. 2020. [↑](#endnote-ref-3)
4. "Box2D." <https://box2d.org/>. Accessed 25 Jan. 2020. [↑](#endnote-ref-4)
5. "Android version history - Wikipedia." <https://en.wikipedia.org/wiki/Android_version_history>. Accessed 25 Jan. 2020. [↑](#endnote-ref-5)
6. "Android Developers." <https://developer.android.com/>. Accessed 25 Jan. 2020. [↑](#endnote-ref-6)
7. "Java | Oracle." <https://www.java.com/>. Accessed 25 Jan. 2020. [↑](#endnote-ref-7)
8. "Handler | Android Developers." 27 Dec. 2019, <https://developer.android.com/reference/android/os/Handler>. Accessed 25 Jan. 2020. [↑](#endnote-ref-8)
9. "EventListener | Android Developers." 27 Dec. 2019, <https://developer.android.com/reference/java/util/EventListener>. Accessed 25 Jan. 2020. [↑](#endnote-ref-9)
10. "SensorEventListener | Android Developers." 27 Dec. 2019, <https://developer.android.com/reference/android/hardware/SensorEventListener>. Accessed 25 Jan. 2020. [↑](#endnote-ref-10)
11. "Factory method pattern - Wikipedia." <https://en.wikipedia.org/wiki/Factory_method_pattern>. Accessed 25 Jan. 2020. [↑](#endnote-ref-11)