Coloristance - Testrapport

1 Introduktion

Coloristance är ett spel där spelaren förflyttar sig genom olika rum och banor med hjälp av nycklar.

1.1 Spelidé

Spelet går ut på att spelar ta sig igenom rum med hjälp av nycklar för att klara olika banor. Nycklarna ligger utplacerade i de olika rummen och dessa kan lagras i ett inventory så att nycklar hittade i ett rum kan användas i ett annat. För att röra sig mellan rummen dras en nyckeln antingen från inventory eller direkt från rumsrutan till dörren med motsvarande färg som nyckeln. På så vis ger dörren access till nästa rum och positionsmarkören i kartan visar spelaren vart den befinner sig. Spelutmaning grundas i att spelaren bara får vistas i ett rum en bestämd tid samtidigt som spelet ställer strategiska krav om vilka nycklar som ska användas i rätt rum för att klara hela banan.

1.2 Generell karaktäristika för applikationen

Spelet kan enkelt byggas ut med mer effekter och ytterligare banor.

2 Tester

Testerna har utförts manuellt.

2.1 Hårdvarumiljö

De manuella testerna utförs på följande fysiska enheter:

- Samsung Galaxy Note 10.1 – Android-version 4.1.2
- HTC Incredible S – Andriod-version 4.0.4
- Sony Xperia Go – Android-version 2.3.7
- Samsung Galaxy S2 - Ansroid-version 4.1.2

Utöver tester på de fysiska enheterna har tester utförts på en Android emulator i operativsystemet Windows 7.

2.2 Mjukvarumiljö

Testerna är utförda på följande operativsystem:

- OS X 10.8.3 (Mountain Lion)
- Windows 7
- Windows 8

Eclipse IDE med inbyggd ADT (Android Developer Tools) har använts tillsammans med Android 4.2 (Emulerad Google API-plattform level 17)

3 Systeminformation

3.1 Systemversion

Coloristance 0.1 beta.

4 Kända buggar och begränsningar

Följande buggar är listade innan utförda tester:

- Musiken startar om då spelaren växlar orientation
- Vid snabb växling av skärmens orientation adderas tid till timern i rummet

5 Testspecifikation

Testerna är specificerade i Requirements-dokumentet (coloristance/doc)

6 Automatiska tester

Vi har använt oss av CodePro-Tools för att generera automatiska tester av hela projektet. Dessa testade koden på annorlunda sätt jämfört med hur vi utförde våra tester och gav därmed mer djup till testningen.

6.1 Kodtäckning (code coverage)

Vi har valt att använda oss av eclEmma som vårt code coverage analysverktyg. Eftersom vi valde att endast använda oss av ett fåtal automatiskt genererade tester i kombination med våra egna, uppnådde vi en code coverage på 19.7%.

6.2 Unit tests

Standard unit-tester används för att testa logiken i denna release.

7 Testrapport

7.1 Funktionstester

Green	Passed test
Red	Did not pass test
Yellow	Not yet implemented

Test ID refererar till Requirements-dokumentet.

TEST ID	Resultat	Kommentar
1	Success	
2	Success	
3	Success	
4	Success	
5	Success	
6	Success	
7	Success	

8	Success	
9	Success	
10	Failed	Not yet implemented