An Intuitive Control API for Catroid

By

Hoo Pei Ying

Collaborating with

Catrobat @ MIT Scratch & App Inventor

Institute of Software Technology, TU Graz

Dissertation submitted in partial fulfilment of

the requirements for the

Bachelor of Technology (Honours)

(Information Communication Technology)

May 2012

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

An Intuitive Control API for Catroid

By

Hoo Pei Ying

A project dissertation submitted to the

Information Communication Technology Programme

Universiti Teknologi PETRONAS

in partial fulfillment of the requirement for the

BACHELOR OF TECHNOLOGY (Hons)

(INFORMATION COMMUNICATION TECHNOLOGY)

Approved by,

____

(Dr. Mohamed Nordin Bin Zakaria)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

May 2012

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

HOO PEI YING

ABSTRACT

In this research, the main objective is to develop an intuitive control API in Catroid to enhance its usability as a graphical programming tool for children and study the human-mobile interaction and experience made possible with this control API. Another objective is to develop this control API in open source development method and benchmark it with the typical software development method.

It would greatly enrich user experience if Catroid can provide support for implementing intuitive control concepts to enhance its usability for children. But currently Catroid do not have control API support to develop intuitive user interaction with the application. In brief, an intuitive control API is missing in Catroid. Without such an API, the potential of Catroid as a programming tool cannot be unleashed.

This research studies the maximization programming power of Catroid and advancement of control API in Catroid into a more intuitive form. This research studies the Open Source Development Model used to develop the control API. The scope of prototype will only covers locating direction, tilting, turning, and shaking motions as the new intuitive control made possible in Catroid

The research methodology is Open Source Development Methodology (OSDM) and the Test-Driven Development Method with Extreme Programming is used for code development. The objective of OSDM is to utilize the online community who is the user and developers of Catroid to review and test source code to improve the software quality.

The intuitive control API where phone sensors are integrated will further improve the user interaction and experience both in using Catroid and its application. The intuitive control API consists of sensor variables and If-Then-Else Command Block. The If-Then-Else Command Block acts as the control and the sensor variables make the control become intuitive. Accelerometer and orientation sensor are implemented in this control API where each of the sensors contributed 3 different

values acted as the sensor variables: X-Sensor Acceleration, Y-Sensor Acceleration, Z-Sensor Acceleration, Azimuth, Pitch, and Roll. These sensor variables can be assigned to or removed from any text field in the Command Blocks using the Formula Editor. The usage of the intuitive control API is simple and straight forward. When a sensor variable is assigned to one of the fields in If-Then-Else Command Blocks, the intuitive control is developed. The Command Blocks in between the If-Statement Command Block and End of If Command Block will be executed whenever the logic condition in the If-Statement is true.

Various intuitive user interactions could be developed depending on the creativity of users. The most popular intuitive user interactions are through locating direction, tilting, turning and shaking motions.

Open Source Development Method allows developers to redefine the user requirements along with the software development which reduce the risk of software failure in the end of development.

ACKNOWLEDGEMENT

I would like to take this opportunity to acknowledge all parties who had contributed to the completion of my final year project. The past one year has indeed been an amazing experience and journey to embark on. I have learnt so many things throughout the completion of my own proposed project.

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Mohamed Nordin B Zakaria for the assistance provided for me to complete my Final Year Project. He has been guiding me from time to time to make sure that there are no complications during the development of my project. Ideas, opinions, and suggestions kept pouring endlessly from him that helped me a lot in developing my project.

I would like to express my utmost and sincere appreciation and gratitude to my collaborator, Catrobat @ MIT Scratch & App Inventor and Professor Wolfgang Slany who gave me the opportunity to contribute to the enhancement of Catroid and provide full technical support and advice to complete my dissertation.

Last but not least, I would like to take this opportunity to thank my family members especially my parents for always being there for me. They have been giving me a great deal of encouragement, advice and support which have led me to the completion of my project. On the other hand, I would also like to thank to my colleagues for the continuous moral and emotional support that they had given me throughout the year.

Thank you.

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Background

Catroid is a new Lego-style graphical programming system of Catrobat that runs on Android devices which is designed for children and novice programmer to create Android mobile application. Catroid is inspired by MIT Scratch and Google App Inventor. (Catroid, 2011). Similar to other programming language, there are many APIs available to build an application in Catroid. Currently, Catroid lacks a controlling and sensor-detecting APIs like what Scratch has. There are only seven two-dimensional surface gestures or touch screen motions available for the control function in Catroid.

API, Application Programming Interface is a set of routines, protocols, and tools for building software applications. With current APIs in Catroid, children could not do higher-level programming with Catroid. A good API makes it easier to develop a program by providing all the building blocks. The interface of API in Catroid is referred to as *Command Blocks* in the *Block Palette*.

The Catroid's target users are novice programmer and children, thus, the users of the projects created by Catroid are novice programmer and children as well as the normal mobile application users. Due to the popularity of intuitive control in mobile application and the concern that Catroid's projects run on mobile devices, it is especially important to have control function in Catroid's project as intuitive as possible for mobile application users. However, currently Catroid does not have API that could help the Catroid's users to build intuitive control function in their project.

Given the popularity of games and animation, it is desirable to have intuitive user interaction in application created by Catroid but currently no API has support to develop the simple intuitive user experience as the example below in their Catroid's project.

"Imagine you are a 10 year old boy. You have a Sprite, which is an Airplane. If you tilt the phone to the right, you have the plane to fluently move to the right side of the screen. If you tilt it to the left, you have the plane to go to the opposite direction." (Catroid, 2011)

To qualify as a higher-level programming language for children, there is a need to enhance the Control APIs instead of just relying on the seven multi-touch/touch motions. The researches and studies prove that sensors are widely used in most of the mobile applications in Android devices to create the intuitive control in application. Hence, the integration of phone sensors in Control API could help children improve the interactivity of their application created in Catroid.

1.2 Problem Statement

Catroid is a new programming tool for children. However, an intuitive control API is missing in Catroid. Without such an API, the potential of Catroid as a programming tool cannot be unleashed. In fact, it would greatly enrich user experience if Catroid can provide support for implementing intuitive control concepts to enhance its usability for children. But currently Catroid do not have control API support to develop intuitive user interaction in application.

1.3 Objectives

The objectives of this project are as follows:

- To implement an intuitive control API in Catroid.
- To study the user interaction and experience made possible with this control API.
- To develop the software using an open source development model, and to benchmark the methodology with typical software development model.

1.4 Scope of Study

This research studies the maximization programming power of Catroid and advancement of control API in Catroid into a more intuitive form. The intuitive control API is implemented in Catroid as *Command Block* with integration of different phone sensors to help children to program user interaction in Catroid application.

This research studies the Open Source Development Model and adopts it to develop the control API. The development method, tools and license used in Open Source Development Model have to be studied to benchmark its pros and cons with typical software development model in this research project. The specification and definition of open source license have to be clear to developers because Catroid is open source software and different open source libraries used in the software development.

Due to expenses and duration of pilot study within 3 months, the prototype only covers locating direction, tilting, turning, and shaking motions as the new intuitive interaction made possible in Catroid.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This section includes research and information that is relevant and necessary to my understanding of the subjects to be discussed in the process of creating the intuitive control API in open source software - Catroid developed using Open Source Development Model to run on Android platform.

2.2 Definition of Intuitive in Control

IUUI research group in Germany defines 'intuitive use' as "*A technical system is intuitively usable if the users' unconscious application of prior knowledge leads to effective interaction"* (Mohs, Hurtienne, Israel, Naumann, Kindsmüller, Meyer & Pohlmeyer, 2006:130).

Two primary input modalities commonly supported by smart phones application are touch screen display and a set of motion sensors. The motion sensors are accelerometers, gyroscopes, orientation sensors (vs. gravity), etc. The two inputs recognized by these devices are different types of gestures. Users can input gestures on the device in two dimensions, using the touch screen of the smart phone as a mobile surface computer. These two-dimensional gestures are *surface gestures*. Users can also input gestures by moving the device, in three dimensions, by translating or rotating the device. These three-dimensional gestures are *motion gestures* (Ruiz, et al., 2011).

The study points out that motion gestures are more intuitive than surface gestures in that the user interacts by using the device itself instead of interacting on the device via a finger or hardware button. To create more intuitive sets of motion gesture sets in a smart phone application, it is important to understand the user's unconscious mental model of how motion gestures are mapped onto the device commands leading to effective interaction (Ruiz, et al., 2011).

2.3 Conceptual Elements of Catroid

According to Daughtry (2008), programming is the new literacy where power will soon belong to those who can master a variety of expressive human-machine interactions. Thus, more programming tools have emerged to educate children nowadays on mastering computer programming skills. Most of these tools run on PC except Catroid. As mentioned in Section 2.1, the aim of this project is to enhance the control API of Catroid.

Catroid is an open source (Perens, 1999) Android software that supports Lego-type graphical programming language designed especially for children. Children interact with software in different ways; they are easily distracted, and they have different motivations as compared to adults. They are likely to 'try out' the software to see what they can do. Thus, Catroid has graphical *Command Block* to represent API in other languages where it eliminates the syntax error and runtime error in programming. In addition, programmers can trial and error in using Catroid by instantly compile and run the project to preview the outcome after modified the *Command Blocks*.

The control API in Catroid only has touch motions *Command Block* shown in Figure 1. The user interactions in application are less intuitive because only surface gestures are implemented in the *Command Block* to control the *Sprites*.

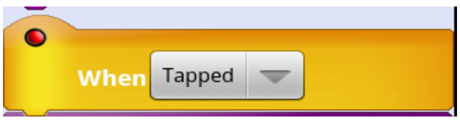


Figure 1: Touch Motions Command Block

Figure 2 is the main menu of the Catroid, when *Current Project is clicked*, *Sprite List* as in Figure 3 appears. After clicking on *Sprite* named "Catroid", Figure 4 which consists of *Scripting Area, Costumes and Sounds* tabs is shown. Programmers can modify the project's code in Scripting Area by dragging and dropping the Command Block. Clicking on the "plus symbol" on top left will show the *Block Palette* shown in Figure 5 while clicking on the "play symbol" will show the *Stage* shown in Figure 7. On block palette, clicking on any category will bring up a list of graphical *Command Blocks* as in Figure 6 where the selected Command Block will add into Scripting Area whenever it is clicked.

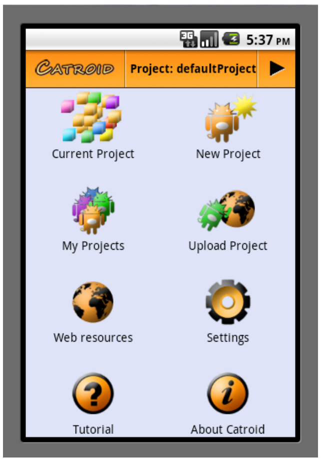


Figure 2: Main Menu

Figure 3: Sprite List

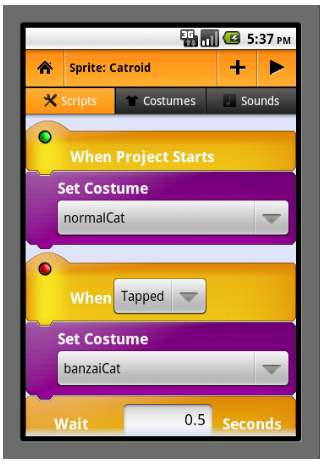


Figure 4:Scripting Area, Costumes, Sounds

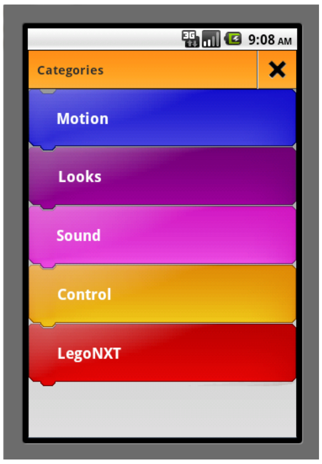


Figure 5: Block Palette

Figure 6: Group of Blocks

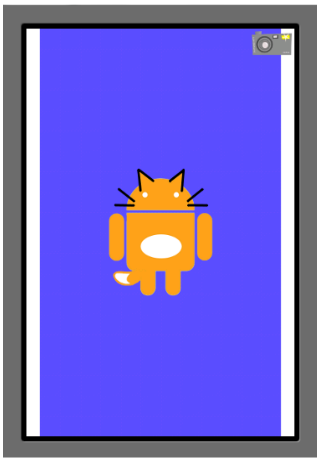


Figure 7: Stage

2.3.1 Children Programming Tools –Control and Concept

According to survey done by Kelleher and Pausch (2005) and Guzdial (2004), Scratch is not the first programming environment and language aimed at novice programmers. Indeed, there is a rich history of different developments for programming tools made for children such as Alice and Greenfoot. Fincher, et al. (2010) and Utting, et al. (2010) have compared Scratch with Alice and Greenfoot and concluded that like Scratch, Alice and Greenfoot are intended to introduce programming to those without prior experience and, as a result, the three systems share many of the same design goals. While all three systems allow media to be imported, Scratch includes tools to draw images and record sounds. According to Maloney, et al. (2010), both Alice and Greenfoot introduce class-based object-oriented programming, and emphasize Java or Java concepts. Greenfoot, since it is Java, also allows students to explore high-performance computation (e.g., complex simulations or cryptography problems) and to extend the system. Alice is the only one of these systems that supports 3-D graphics and it often crashes in middle of programming. Scratch allow the children to create interactive, media-rich projects including animated stories, games, online news shows, book reports, greeting cards, music videos, science projects, tutorials, simulations, and sensor-driven art very easily.

Compared with figures in Section 2.2, Maloney, et al. (2010) claimed that the user interface layout of Scratch (Figure 8) with its prominent command palette and central scripting area, ease children to program. Scratch programming system strives to help users build intuitions about computer programming as they create projects that engage their interests by dragging *Command Blocks* from a palette into the scripting pane and assembling them, like puzzle pieces, to create “stacks” of blocks (Maloney, et al., 2010).

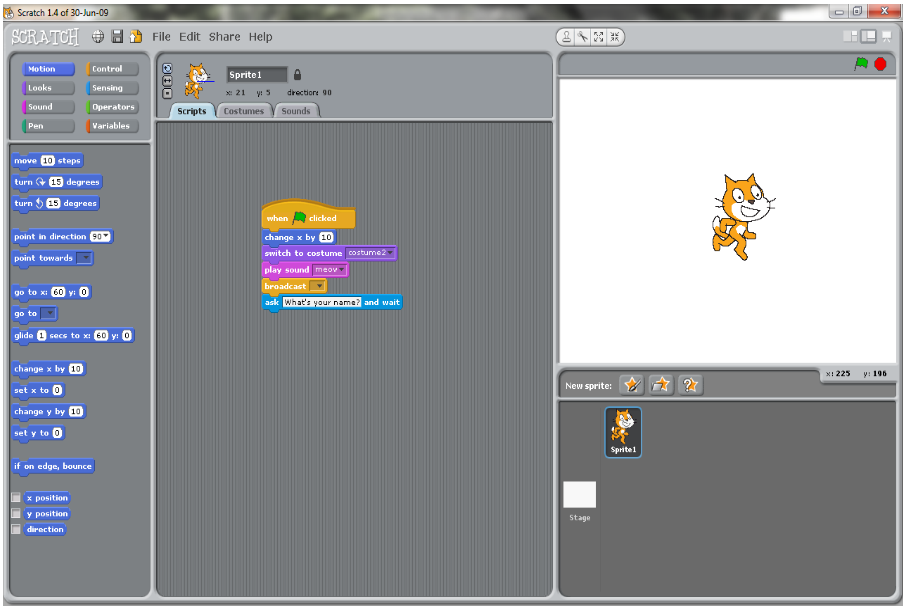


Figure 8: Interface of Scratch

The user interaction for Scratch's application include surface gesture and motion gesture using keyboard, mouse, microphone, game controller and remote sensors. The integration of smart phone sensors in Catroid can reduce short term memory load since some researchers used objects with sensors as an interactive mode to reduce children's cognitive burden to interact with software (Zhou ZY., et al., 2008).

2.4 New Concept of Human-Mobile Interaction

To build the intuitive control API in Catroid, integration of accelerometers with gyros, proximity sensors, or vibratory and shear/torque sensors could greatly enrich the vocabulary motion (Hinckley and Song, 2011). There are several touch+motion techniques that could be implemented on a smart phone which are Tilt-to-zoom, Pivot-to-lock, Hold-and-Shake, Tip-to-select, Soft-vs-Hard-Tap, Swipe-vs-Hard-Drag and Context of touch.

The study shows the users' preference rankings for the techniques in Figure 9. Acording to Hinckley and Song (2011), five of the ten users explicitly mentioned "one-handed" interaction as the best thing about *Tilt-to-zoom*. Users liked how they could combine wrist, finger, and device motion to articulate lightweight interactions with less “friction” in the user interface. Users commented that the techniques were “intuitive and easy to transition to different modes,” “easy and magical,” or that “the icons are alive!”

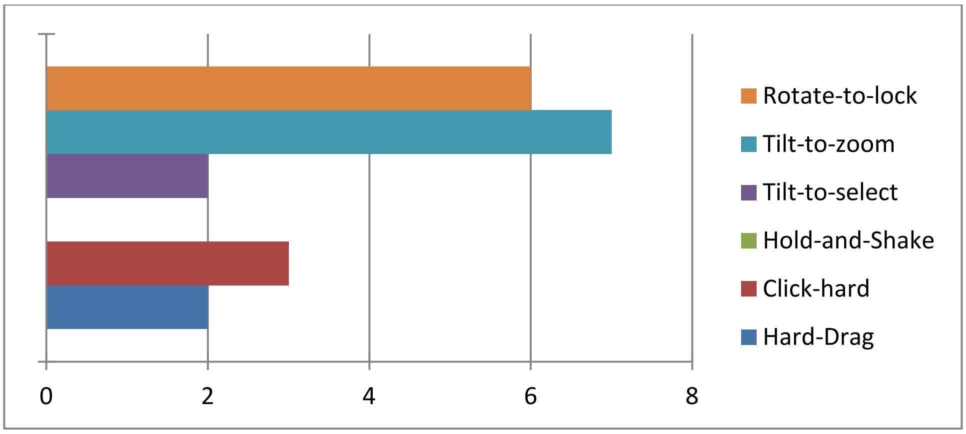


Figure 9: Users' Preference Ranking for the Techniques

Combining motion gestures with touch is a simple idea that has been explored. Hassan (2009)’s “Chucking” technique uses a simultaneous touch+motion gesture to toss a file from a mobile device to a wall display. The user holds a finger on the file’s icon, while indicating where to place the file via a motion gesture. Rahman (2009) also uses touch+motion to measure wrist deflection angles. Motion has also been used as a cue for grip-sensing mobile devices. For example, an accelerometer can trigger implicit grip sensing when a mobile device is held still (Kim, et al., 2006). Graspables use accelerometers to sense the orientation of objects and to trigger grip sensing at the right moment (Taylor, 2009).

However, the touch+motion techniques mentioned above might confuse children as the study of surface gestures for children shows that children tend to make mistakes when doing complex gestures (McKnight and Fitton, 2010). Druin (1999) concludes that children aged 5 –7 want interface interaction that they can easily control but not too simple. The user-defined motion gestures without touch proposed by Riuz, et al. (2011) shown in Figure 10 are more desirable way for children compared to touch+motion techniques.

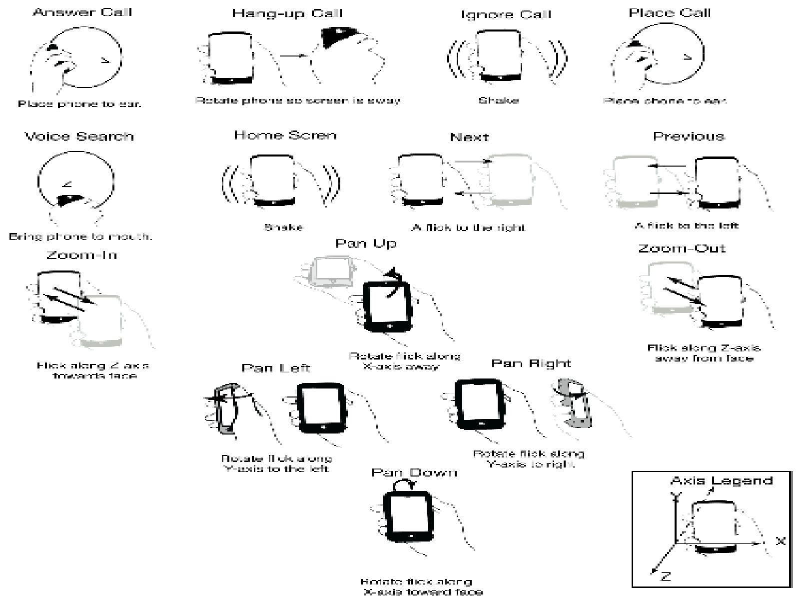


Figure 10: Motion Gestures Sets by Riuz, et al.

Both the studies above concluded that the integration of phone sensors in Catroid could create an intuitive control API for children in a way that reduces children's conscious application of prior knowledge to program an intuitive user interaction.

2.5 Smartphone Operating Systems

Lin and Ye (2009) listed several leading smart phone operating systems in the market which include Apple's iOS, Google's Android, Microsoft's Window Mobile, Nokia's Symbian, RIM's BlackBerry OS and embedded Linux distribution such as MeeGO. Catroid runs on Android operating system. However according to Hall and Anderson (2009), Android is not the only smart phone that has the potential to solve the dissatisfaction that users have with their devices. Hence, Catroid could be possible developed in other smart phone OS as listed above to have intuitive control API.

According to Gartner (2011), by the end of 2011, Android will move to become the most popular operating system worldwide and will build on its strength to account for 49 percent of the smart phone market by 2012 with Apple's iOS remaining as the second biggest platform. Apple's iOS and

Google's Android are the preferred platforms to go to. But Lin and Ye (2009) said the iPhone is a closed smart phone running a closed OS, which means Apple has the full control on its hardware and software. Hence, the open-platform nature in Android is more desirable for Catroid. Android runs on a Linux-based architecture with Java applications running on top of it is illustrated in Figure 11. Android has free Android Software Development Kit provides the tools, APIs libraries, tutorials and demo source codes necessary to develop applications unlike Apple iOS required registration fees to use the development tools and the forum.

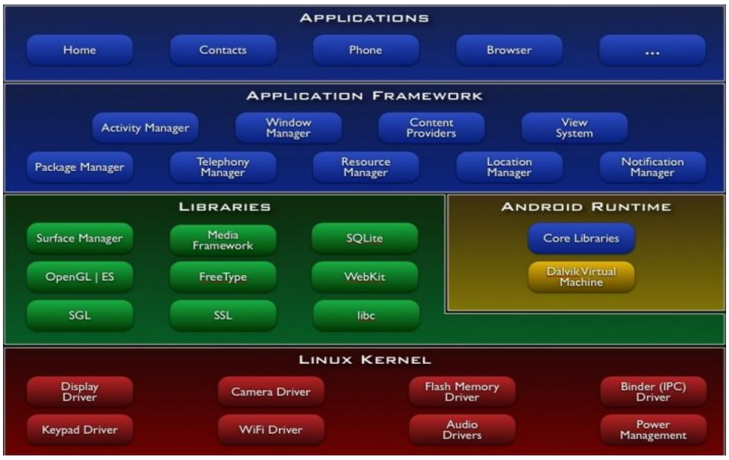


Figure 11: Android Architecture

2.6 Catroid Development Tools & Methods & License

According to writer Martin, et al. (2009), writing clean code is what a developer must do to be called as a professional. To eliminate flawless codes, the whole Catriod development is using 100% test-driven development method where it uses automated tests and Robotium test to reduce resources usage in debugging. For code quality assurance and code refactoring flexibility, writing high quality code is the only documentation for Catroid developers across worldwide, with a bunch of other practises from extreme programming and usability engineering. Catroid team created rules of coding as shown in Appendix 1 based on the book written by Martin, et al. (2009) which results in more efficiency in maintaining the codes without breaking anything.

2.6.1 Development Tools

Catroid is an Android application. Any Java IDE tool can be used to develop open source Android applications. One of the most popular Android application development platforms is Eclipse IDE because it can run on any OS platform and supports many plugins including Android ADT, Open GL ES, Mercurial, Mylyn, Oracle, Apache, UMLet, JSP and etc. Android Development Tools (ADT) can be installed in Eclipse IDE to extends the capabilities of Eclipse to quickly set up new Android projects, create an application UI, debug applications using the Android SDK tools, and even export signed (or unsigned) APKs in order to distribute the application. In general, developing in Eclipse with ADT is a highly recommended approach by Google's Android and is the fastest way to get started with Android.

In Open Source Software development, the participants, who are mostly volunteers, are distributed among different geographic regions. Tools are needed to aid participants to collaborate in the development of source code. In this case, Concurrent Versions System (CVS), Subversion revision control system (SVN) and distributed revision control systems such as Git and Mercurial are useful. These revision control systems help manage the files and codes of a project when several people are working on the project at the same time. This is done by moving the file into the users' directories and then merging the files when users have committed the changes. Revision control systems also enable one to easily retrieve a previous version of a file which is very useful in collaborative programming environment. (Open source software development, 2011). Catroid implemented Mercurial in the beginning then moved to Git recently because Git provides more features and functions such as revert and track of progress in branches.

Most of open source software is stored and made publicly available on code sharing and hosting platforms such as Google Code Hosting, Github.com, Sourceforge.net, Codeplex, and Bitbucker which enable designers and developers to share their code with the web community. The platform also serves as backup for the codes. Github.com is used in this research.

2.6.2 Open Source Development Method

Software or application is developed based on a single or combined software development process model. There are many software process models in software engineering; the most popular models are V model, Prototyping Model, Increments and Iterations Model, and Agile Model (Pfleeger & Atlee, 2010). Open Source Development methodology is not the newest but the most popular method adopted in smart phone application development.

Most of the software developed in Open Source Development Model (OSDM) is open source. Many mobile applications are open source and the mobile application developers prefer to adopt OSDM in their application development. The source code of open source may be freely modified and redistributed without charge or limitations on modifications to encourage collaborative development such as Linux OS, Apache Server and Mozilla Browser. OSDM emphasizes faster development and lower overhead, as well as closer user relationship and exposure to a broader market. In OSDM, when the software is developed under a model of systematic peer-review, it can be incrementally improved and more easily tested, leads to innovation and rapid advancement resulting in a highly reliable product (LONCHAMP) because more people looking at the code will results in more “bugs” found (Raymond, 2001).

Typically, OSDM adopts the Hybrid process models. All of the Agile methods are in essence applicable to OSDM, because of their iterative and incremental character (Open source software development, 2011). Catroid is an open source that adopts OSDM, thus this project will adopts OSDM with extreme programming development methodology.

2.6.3 Open Source License

A license defines the rights and obligations that a licensor grants to a licensee. There are several licenses compatible for open source software such as GNU General Public License, Creative Common Attribution License, GNU

Lesser General Public License and etc. This project adopts GNU AFFERO GENERAL PUBLIC LICENSE as published by the Free Software Foundation, either in version 3 of the license or any later version in future. The complete copy of the original License can be referred to Appendix 3. The benefit of using this license is that it acknowledges the hard work of the contributors while leaving the source free for modification and redistribution.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This project applies Extreme Programming Process Model in Agile Development method and licensed as open source software. The timeline and key milestones for this project are shown in Tracking Gantt Chart in Appendix 2.

3.2 Previous Related Work

Several published works contain different sensor design and framework to develop application or software in different devices. The architectures of their framework were illustrated in diagrams and pictures as a guideline to help the developers smoothly implemented codes and traced back the relationship between the classes and function calls.

In Android Development Reference Library, the APIs that are needed to develop intuitive control API in Catroid are in Package android.hardware.

3.3 Research Methodology

3.3.1 Open Source Development Method

The Figure 12 (M.Abbing, 2006) below shows the development process cycle that this project is going to apply based on Extreme Programming. The existing project that will be further enhanced is Catroid.

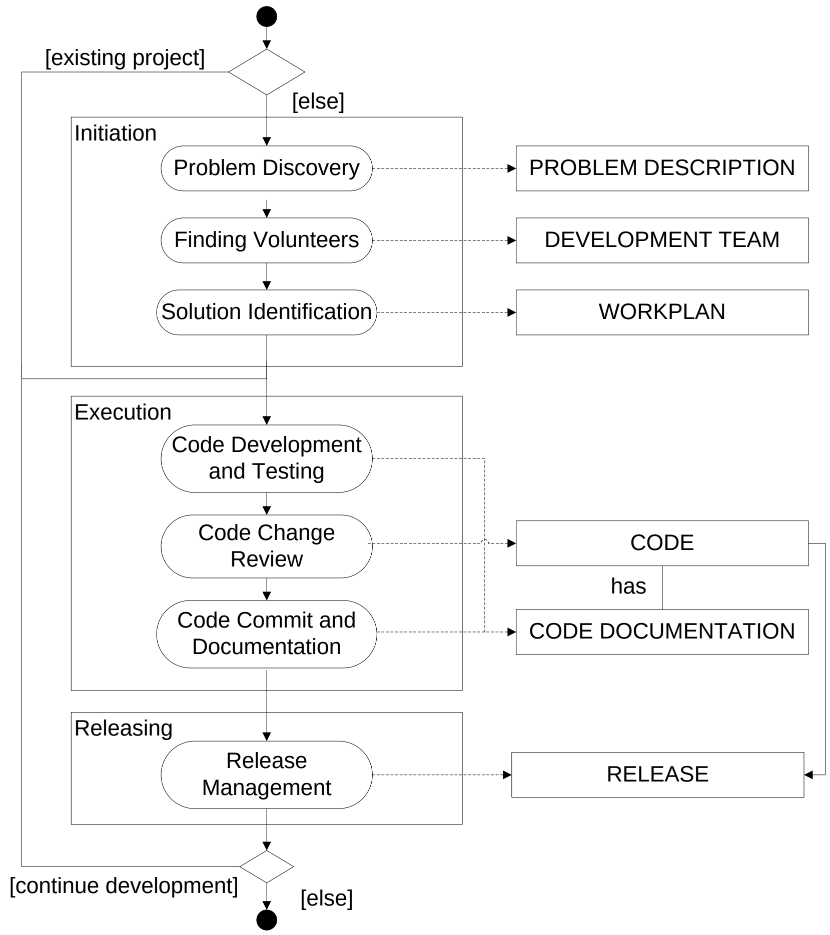


Figure 12: Open Source Development Model

In open source development model, first phase is initiation started by identifying existing projects to be further improved and the problem in existing projects. It is discovered that intuitive control features is missing in Catroid. The identified problems and ideas are being posted online in Google Code Hosting (Appendix 4) and Android community website to find volunteers to form the development team. This project managed to get Software Technology Department in TU Graz, Austria who is the founder of Catroid to support and provide remote assistance. Solution is found after several times of discussion and ideas refining. A work plan which is this proposal is made for outlining the important elements in developing this project.

The next phase is execution which is the development and testing of codes. Any java class that contained core functionality of the project must have its own test class to ensure consistency and continuation of codes. The code is released when it passes the code review phase. This review phase is very important to eliminate flaw in code and for benchmark evaluation. The accepted code is committed and documented properly before deploy to the world.

The last phase is software release. The first software deployment is the first prototype. The first prototype in this project is given to a selected target group to conduct project evaluation and redefine the user requirements. After tested by targeted user, bugs and errors are identified and the project development phase is iterated back to either initiation for planning or execution for coding. Open source software is long term sustainable developing software, there is no end for the development because the software is developed together with the users. Development is stopped only when the software is no longer used by the public.

3.3.2 Test-Driven Development Method

Figure 13 is the flow diagram of Test-Driven Development Method. The Test-Driven Development Method is used in code development phase to ensure that the application is written for testability.

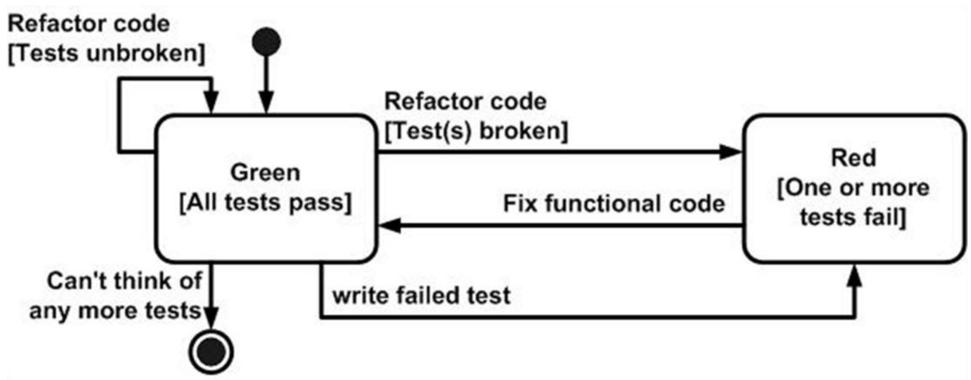


Figure 13: Test-Driven Development Method

Test-Driven Development Method relies on the repetition of a very short development cycle: to add a new functionality, first the developer writes the failing automated test cases which are the Junit Test and Robotium Test that defines a desired improvement or new function, then produces code to pass that test and finally refactors the new code to acceptable standards.

3.4 Technical Specification Design

In the Figure 14 below shown the flow of logics how the control API works. Firstly, block of control API which implements the sensor is added into script and a condition is set. Then the accelerometer and digital value changes when the Android device is tilted or manipulated. Sensor is triggered and immediately the respective script is executed after the condition is checked. The effect and respond are the outputs in the display screen for the view of user. After the sensor event is cleared, it is ready to intercept any input event.

Within the scope, only tilting, turning, shaking motions and locating direction will be designed and implemented with integration of phone sensor in *Command Blocks*. These motions will be feasible to conduct with "one-handed" interaction.

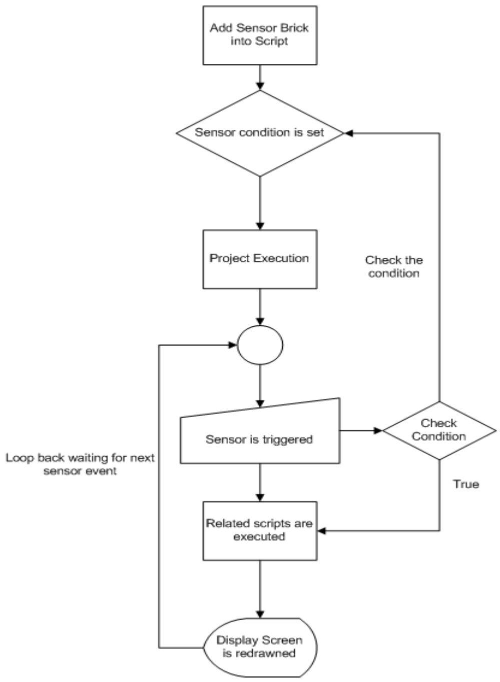


Figure 14: Flow Diagram of Control API Logic

In the designing process of intuitive control API, interface design rules by Schneiderman's Eight Golden Rules can be referred: Strive for consistency, Enable frequent users to use shortcuts, Offer informative feedback, Design dialogues to yield closure, Offer error prevention and simple error handling, Permit easy reversal of actions, Support internal locus of control and Reduce short term memory load. These rules will guide the designers to design the intuitive control API that is consistence, easily understand, no syntax error, easy handling and user-friendly.

3.5 Technology Used

List of technology for this project:

- Android SDK 1.6 r1
- Android Development Tools v9
- Eclipse Helios Sr2
- GitHub.com/ Google Code Project Hosting

Below is the knowledge required for this project:

- Object-Oriented Programming (Java)
- Android Development Framework

The development environment is Eclipse Helios IDE for Java EE developers. Integrating the SDK with Eclipse simply required a download of the Android Development Tools (ADT) Plugin (Android SDK).

The Android platform is specifically designed for Java as it is available on many different platforms. This project used Object-Oriented Programming style of development for easier code maintenance. Google's Android has provided comprehensive documentation to assist in the development process. To get started, the Android SDK and Eclipse ADT plug-in were required (Android SDK). The Android SDK comes with an emulator that simulates a clean install of an Android device. Multiple screen resolutions are available for testing layouts, as well as an interface to connect actual Android devices. For this project, a real Android device is needed for testing because the emulator does not support sensors. GitHub.com and Google Code Project Hosting are used to store the prototype source code for source code sharing and as a backup.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Deliverable's Interface

The intuitive control API is the combination of sensor variables and If-Then-Else Command Block. The If-Then-Else Command Block acts as the control and the sensor variables make the control become intuitive.

4.1.1 Sensor Variables

Two sensors are being integrated into Catroid: The Accelerometer and The Orientation Sensor. The Accelerometer senses the Android device's accelerometer, which detects measures acceleration in three dimensions. Acceleration is measured in SI units (m/s^2). If the device is at rest lying flat on its back, the Z acceleration will be about 9.8. When it is being lifted, it produces three values:

- **X-Sensor Acceleration**: Positive when the device is tilted to the right (that is, its left side is raised), and negative when the device is tilted to the left (its right size is raised).
- **Y-Sensor Acceleration**: Positive when its bottom is raised and negative when its top is raised.
- **Z-Sensor Acceleration**: Positive when the display is facing up, and negative when the display is facing down.

The orientation sensor determines the phone's spatial orientation. An orientation sensor is a non-visible component that reports the following three values, in degrees assume that the device itself is not moving:

- **Azimuth**: 0 degree when the top of the device is pointing north, 90 degrees when it is pointing east, 180 degrees when it is pointing south, 270 degrees when it is pointing west, etc.
- **Pitch**: 0 degree when the device is level, increasing to 90 degrees as the device is tilted so its top is pointing down, then decreasing to 0 degree as it gets turned over. Similarly, as the device is tilted so its bottom points

down, pitch decreases to −90 degrees, and then increases to 0 degree as it gets turned all the way over.

- **Roll**: 0 degree when the device is level, increasing to 90 degrees as the device is tilted up onto its left side, and decreasing to −90 degrees when the device is tilted up onto its right side.

The values of the two sensors are implemented as the variable/parameter named X-Sensor, Y-Sensor, Z-Sensor, Azimuth, Pitch and Roll These variables can be assigned to or removed from any text field in the Command Blocks using the Formula Editor as in Figure 15. The Formula Editor has the current editing Command Block and an editor textbox on top, 3 buttons at the centre and the keypad at the bottom.

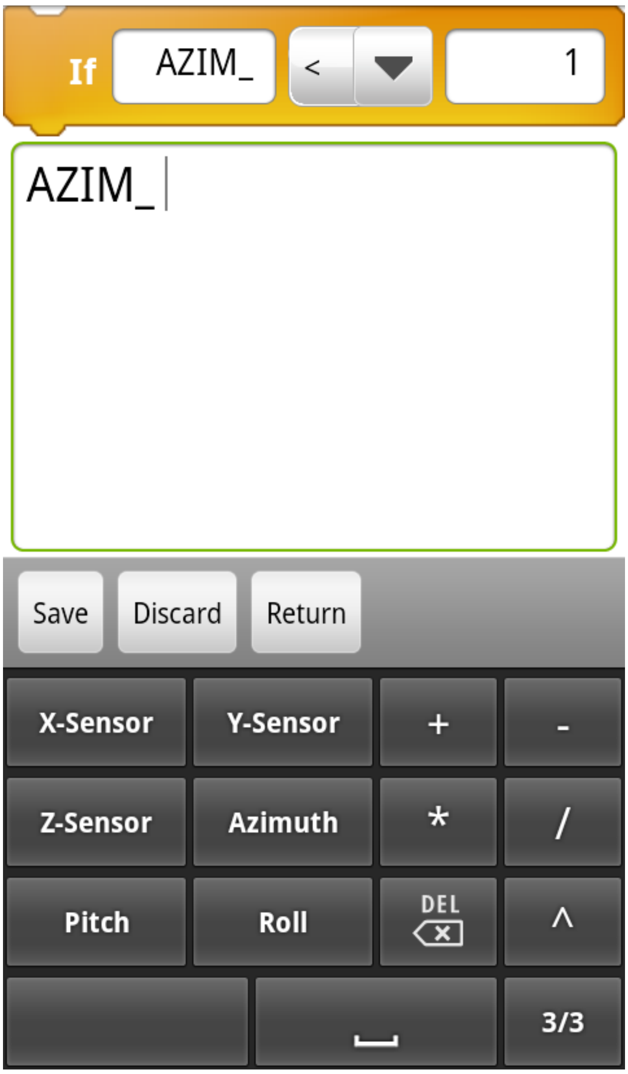


Figure 15: Sensor Variables in Formula Editor

By clicking on the text field of any Command Blocks in Scripting Area, the Formula Editor appears. In the Formula Editor, clicking on one of the 6 sensor variables will append the respective variable identifier in the text field and edit text box. Then, either "Save" button must be clicked to confirm the changes made to the text field or "Discard" button is clicked to discard the changes before clicking on "Return" button. Otherwise, the warning message will be popping out to remind the user as in Figure 16.

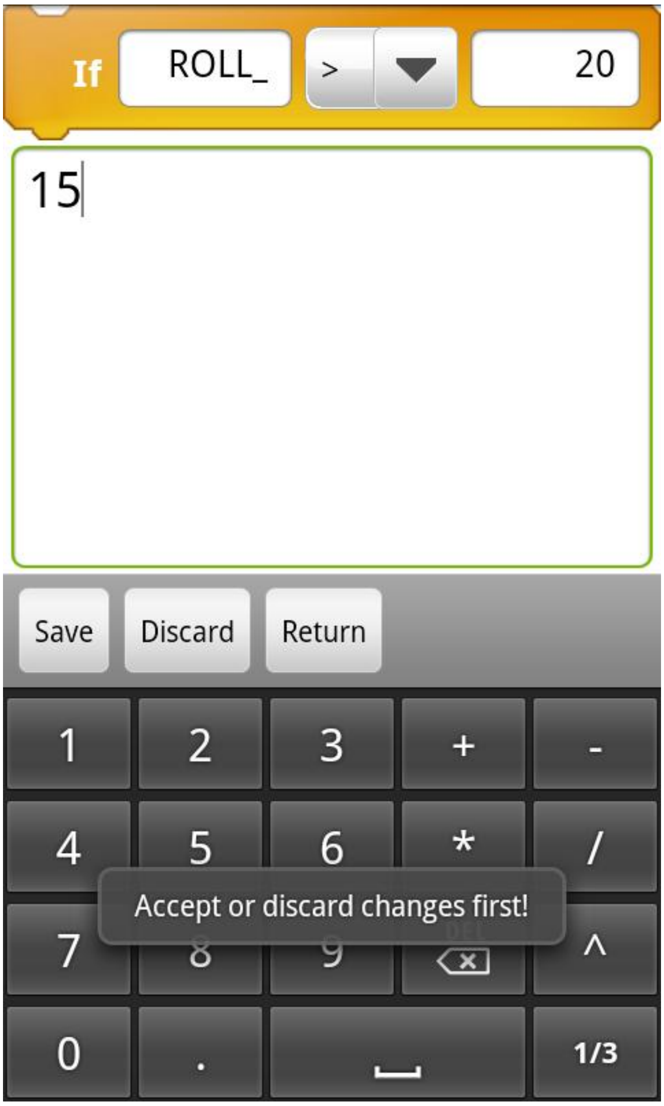


Figure 16: Warning Message to Accept Changes

The layout of Formula Editor and its operation are implemented according to 6 rules in the Schneiderman's Eight Golden Rules: The showing of the Command

Block on top reduces short term memory load and the instant changing of text field value offers informative feedback. The 3 buttons offers error prevention and simple error handling, permits easy reversal of actions and has design dialogues to yield closure. The whole layout design supports internal locus of control of the user.

In addition, the sensor variables can be included in an equation and calculated in the Formula Editor as in Figure 17. Besides mathematic operator, there are also basic mathematic functions to be used in the equation.

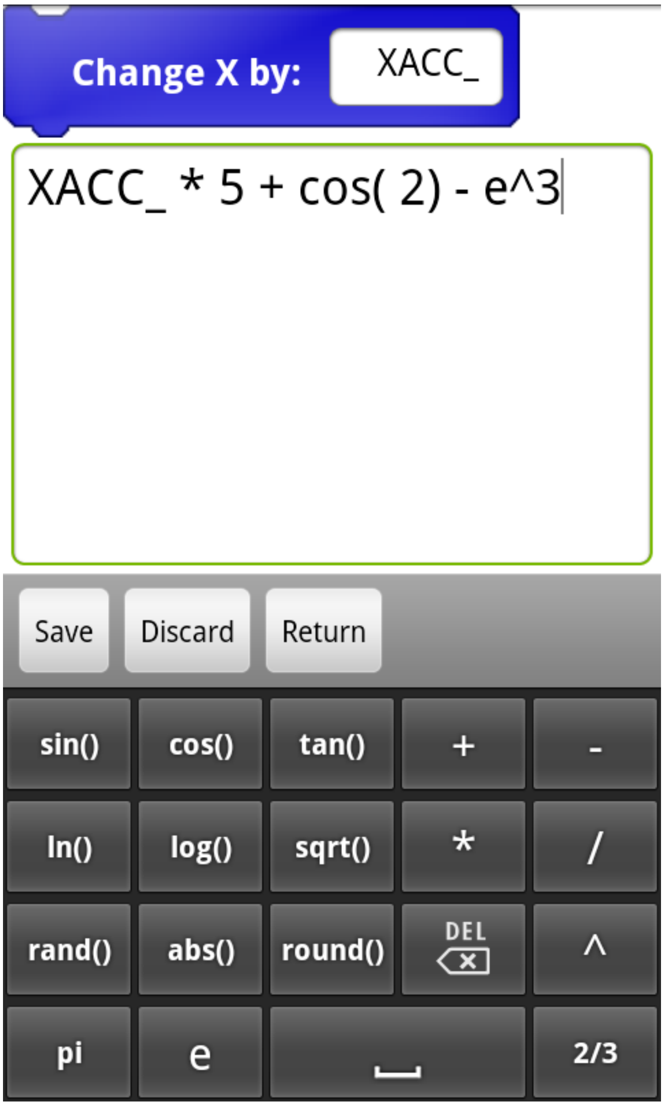


Figure 17: Equation in Formula Editor

4.1.2 If-Then-Else Command Block

The Figure 18 is the interface of the If-Then-Else Command Block to be used together with sensor variables to create intuitive control. The layout of the If-Then-Else Command Block is following the outlook of Control Categories to strive for consistency and it has 2 text fields and a logic operator.

Figure 18: If-Then-Else Command Block

4.2 The Mechanism of Intuitive Control API

The usage of the intuitive control API is simple and straight forward. When a sensor variable is assigned to one of the fields in If-Then-Else Command Blocks, the intuitive control is developed as in Figure 19. The Command Blocks in between the If-Statement Command Block and End of If Command Block will be executed whenever the logic condition in the If-Statement is true.

Figure 19: Intuitive Control API

4.2.1 Sensor Coordination System

In general, the sensor framework uses a standard 3-axis coordinate system to express data values as shown in Figure 20. For most sensors, the coordinate system is defined relative to the device's screen when the device is held in its default orientation. When a device is held in its default orientation, the X axis is horizontal and points to the right, the Y axis is vertical and points up, and the Z axis points toward the outside of the screen face. In this system, coordinates behind the screen have negative Z values.

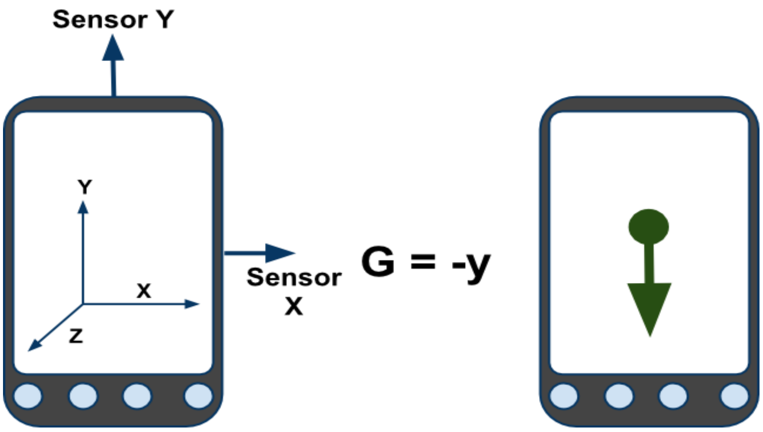


Figure 20: Default Sensor Coordination System

The most important point to understand about this coordinate system is that the axes are not swapped when the device's screen orientation changes—that is, the sensor's coordinate system never changes as the device moves. The Android sensor APIs define the sensor coordinate space to be relative to the top and side of the device — not the short and long sides. When the system reorients the screen in response to holding the phone sideways, the sensor coordinate system no longer lines up with the screen's coordinate system as in Figure 21, unexpected errors are generated in the intuitive control Command Blocks. Originally, X axis refers to Sensor X and Y axis refers to Sensor Y. But, it changes to X axis refers to Sensor Y and Y axis refers to Sensor X after the coordination system reoriented.

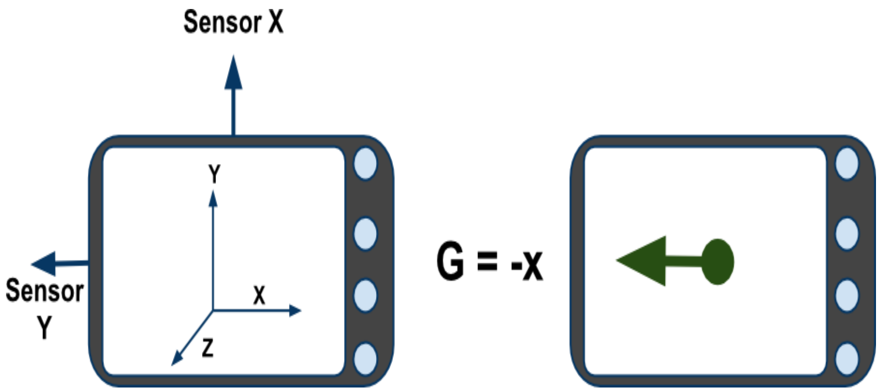


Figure 21: Reoriented Sensor Coordination System

When the sensor variable is used in a Catroid project, the value of Sensor X and Y will be inverted if the orientation of phone changed, thus creating unexpected error. This reduces the usability of the intuitive control Command Blocks and the users will need to take into account this complicated coordination issue whenever they do programming in Catroid. Therefore, the orientation check is implemented to check and swap the reference of the sensors automatically behind the codes to reduce the memory load of users and create usability.

4.3 The Intuitive User Interaction & Experience

The limitation of the current control API in Catroid is eliminated where different phone sensors are integrated to support children to develop new intuitive human-mobile interaction in Catroid. The intuitive control API can program various intuitive user interactions in Catroid depending on the creativity of the programmer. The intuitive user interactions which in common use are shaking, locating direction, titling and turning motion. These four user interactions can be developed according to the script examples below.

The Figure 22 below is the script example to detect shaking motion using X-sensor Acceleration value. The script tells that the particular Sprite will keep rotating anti-clockwise if a user do a shaking motion with the device.

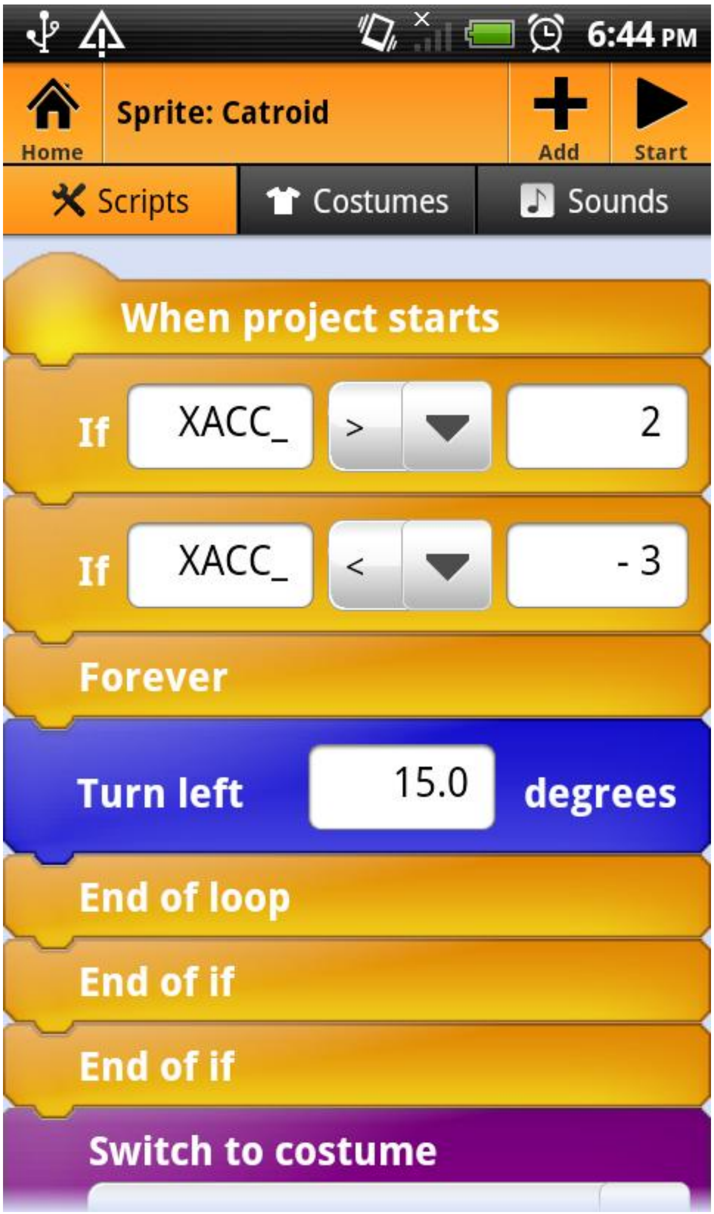


Figure 22: Shaking Motion

The Figure 23 below is the script example to detect direction using Azimuth angle value. The script tells that the particular Sprite will speak out the direction which the top of the device is pointing.

Figure 23: Locating Direction

The Figure 24 below is the script example to detect tilting/ turning/ rotating motion using Roll angle value. The script tells that the particular Sprite will move to the left if the right side of the device is raised and vice versa.

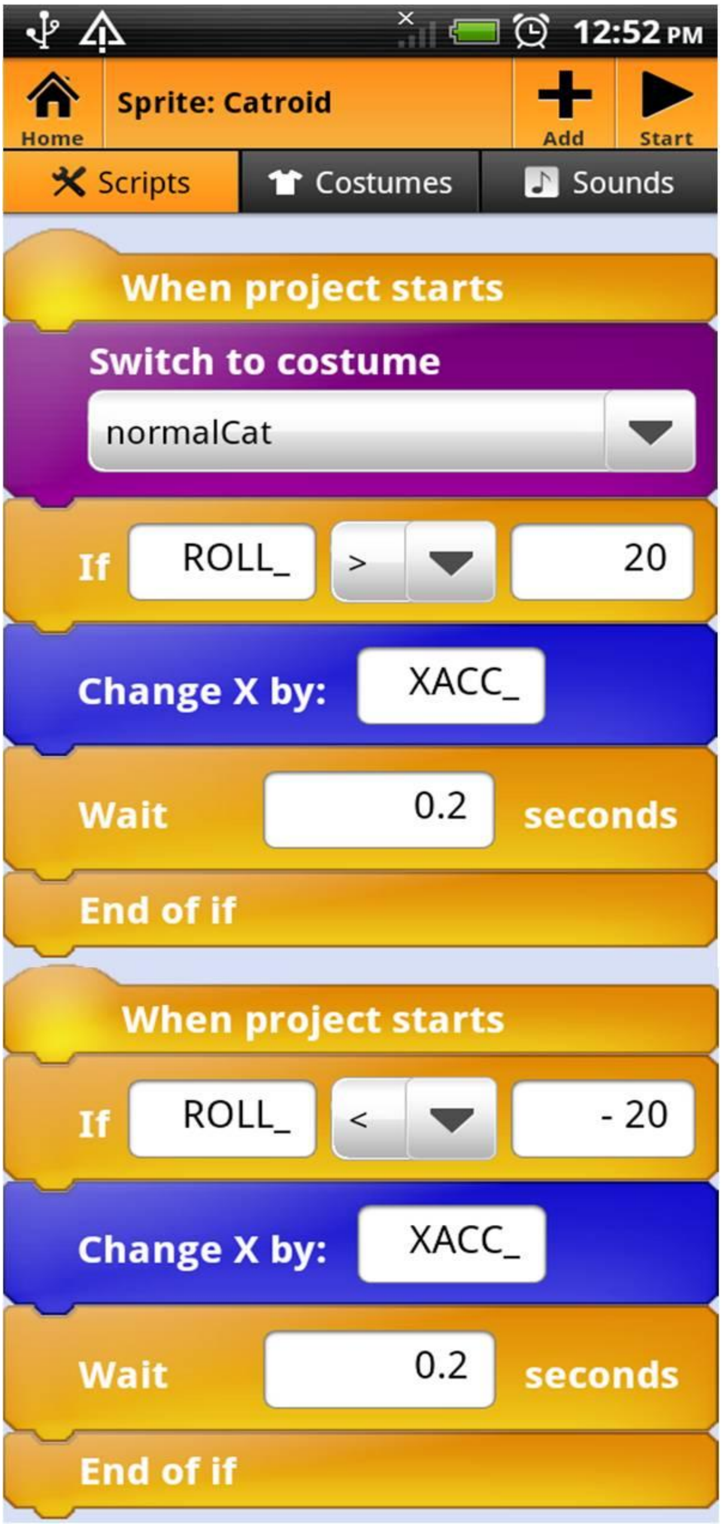


Figure 24: Titling/Turning/Rotating Motion

4.4 Open Source Development Method Benchmarking

The open source development method (OSDM) is different than typical software development method. Its characteristics and good practises can improve code quality, communication, effectiveness, and performance that typical software development method could not achieve.

The OSDM maintains a source code tree that is open and available for all to see and access. This allows full transparency and extensive peer review where all members of the community can comment and offer suggestions and bug fixes. Making the code public without having a fully working version helps the project to redefine user requirements even in the middle of development. Subsequently, the risk of project failure is reduced.

In addition, the OSDM allows this project to make a release available early to be used by the user community and then update the release as the software is modified The "release early release often" practice in OSDM gives a good estimate of the progress and help catch bugs early. It applies small incremental changes in release to make the source code easier to understand and test. Since this project is collaborating with 50 developers, the source code must be easy to understand and test after changes are made.

Usually, the OSDM promotes developers to use open source software and build on top of it to improve efficiency and increases cost savings. The OSDM even foster code reuse practice since it encourages developers to build reusable software components if there is a need to develop from scratch.

The OSDM uses Bottom-up-development approach where project members who done the most work get the most say when it comes to making design and implementation decisions while the typical software development method uses Top-down-development where project management makes the decision and pushes it down to the implements. This approach helps to motivate the developers to contribute more.

The open source development team primarily works together in a decentralized fashion with little hierarchy thus it is essential to have Bug tracking system, Automated test cases, Patch tracking system, and Revision Control System to help monitoring the progress of the team. OSDM welcomes code contributions written by volunteers, however the open source community takes security very seriously and any development or capability that jeopardizes the security of the software is flagged and not included in the software until the security concern is dealt with. In this project, Github is used as the patch tracking system and revision control system. Github is very useful to revert back to the unbroken state and discard all the changes made in the code. This saves time because debugging is time-consuming.

Although there is no formal documentation, the open source community follows a strict coding style to make it easier to understand the code, review it, and revise it quickly. Unclear and messy codes would not be accepted and pushed into the code tree by the community. Furthermore, test projects are created for large open source projects to create test suites and automate testing such as Junit Test and Robotium Test. The strict coding style and test projects can help to enhance the security of the software and ensure the compatibility of the new feature implementation.

CHAPTER 5

CONCLUSION & RECOMMENDATION

5.1 Conclusion

This project eliminates the limitation of the current control API in Catroid to become a higher-level programming language for children. Currently, children can only program the user interaction in application with lower-level programming due to the limited functions of the current Catroid control API. With the intuitive control API where phone sensors integrated, children can develop an interactive application easily in Catroid.

The new intuitive control API not only benefits the children as a programmer but also the users who use the application programmed by them. The Catroid has a huge potential to become the next premier graphical programming tool not only for children and novice users of all ages but a new powerful programming tool for academic usage in computer science education.

An intuitive control API consists of sensor variables and If-Then-Else Command Block. The If-Then-Else Command Block acts as the control and the sensor variables make the control become intuitive. The sensor variables are implemented with the integration of Accelerometer and Orientation sensor. The accelerometer has 3 values: X-Sensor Acceleration, Y-Sensor Acceleration and Z-Sensor Acceleration while Orientation sensor has another 3 values: Azimuth, Pitch and Roll. The sensor variables can be assigned to and removed from Command Blocks using the Formula Editor. The intuitive control is created when the sensor variable is assigned to one of the text fields in If-Then-Else Command Blocks. In addition, the value of the sensor variables can be used in equation and calculation.

The interfaces of sensor variables, If-Then-Else Command Block and Formula Editor are designed referring to the interface design rules by Schneiderman’s Eight Golden Rules.

The usage of the intuitive control API is simple and straight forward. When a sensor variable is assigned to one of the fields in If-Then-Else Command Blocks, the intuitive control is developed. The Command Blocks in between the If-Statement Command Block and End of If Command Block will be executed whenever the logic condition in the If-Statement is true. There are various intuitive user interactions in Catroid can be programmed using the intuitive control API depending on the creativity of the programmer. The intuitive user interactions which in common use are shaking, locating direction, titling and turning motion.

This project chose to adopt open source development method to benchmark it with the typical software development method. The open source development method helps to produce higher-quality software with lower cost. Its characteristics and good practises can improve code quality, communication, effectiveness, and performance that typical software development method could not achieve. Besides, the open source development method reduces the risk of the software failed meet the user requirement.

Overall, this project has a significant impact to the programming power of Catroid and its user experience. It makes intuitive user interaction applicable for children to use in their application. This project will bring the programming language for children into the next higher level, subsequently improving usability to novice programmers, both children and adults.

5.2 Recommendation

There are several recommendations that could be considered to make this project better.

The first recommendation is the naming of the sensor variables. The selection of words and language are important for children to understand how the control API works. It will be better if icons or pictures are used to illustrate the functions of control API.

The second recommendation is to create a tutorial or video to demonstrate the usage of the intuitive control API. Besides, some sample Catroid projects that contain different intuitive control should be included in Catroid setup.

The third recommendation is to develop more sensor variables such as Light Sensor to detect ambient light, Temperature Sensor to detect surrounding temperature and Pressure Sensor to detect pressure.

REFERENCES

Android SDK | android developers Retrieved 11/16/2011, 2011, from http://developer.android.com/sdk/index.html

Catroid - an on-device visual programming language for android inspired by scratch - google project hosting Retrieved 11/16/2011, 2011, from http://code.google.com/p/catroid/

Daughtry III, J. M. Programming, kids, collaborating, and communities.

Druin, A., Ed. (1999). The Design of Children's technology, Morgan Kaufmann Publishers, Inc.

Fincher, S., Cooper, S., Kölling, M., & Maloney, J. (2010). Comparing alice, greenfoot & scratch. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, pp. 192-193.

GNU affero general public license - GNU project - free software foundation (FSF) Retrieved 11/16/2011, 2011, from http://www.gnu.org/licenses/agpl.html

Gartner says android to command nearly half of worldwide smartphone operating system market by year-end 2012 Retrieved 12/5/2011, 2011, from http://www.gartner.com/it/page.jsp?id=1622614

Guzdial, M. (2004). Programming environments for novices. In S. Fincher and M. Petre (Eds.), *Computer Science Education Research* (pp. 127-154). Lisse, The Netherlands: Taylor & Francis.

Hall, S. P., & Anderson, E. (2009). Operating systems for mobile computing. *Journal of Computing Sciences in Colleges, 25*(2), 64-71.
Hassan, N., Rahman, M., Irani, P., Graham, P. Chucking: A One-Handed Document Sharing Technique. *INTERACT'09*.

Hinckley, K., & Song, H.,Y. 2011. Sensor synaesthesia: touch in motion, and motion in touch. In Proceedings of the 2011 annual conference on Human factors in computing systems (CHI '11). ACM, New York, NY, USA, 801-810. DOI=10.1145/1978942.1979059 http://doi.acm.org/10.1145/1978942.1979059

Kelleher, C. & Pausch, R. (2005). Lowering the barriers to programming: a taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys, 37*(2), 88-137.

Kim, K.-E., et al. Hand Grip Pattern Recognition for Mobile User Interfaces. *Proceedings of AAAI/IAAI-2006: Innovative Applications of Artificial Intelligence*. 2006.

Lin, F., & Ye, W. (2009). Operating system battle in the ecosystem of smartphone industry. *Information Engineering and Electronic Commerce, 2009. IEEC'09. International Symposium on,* pp. 617-621.

Lonchamp, J. (2005). Open source software development process modeling. *Software Process Modeling,* , 29-64.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 16.

Martin, R. C., Feathers, M. C., Ottinger, T. R., Langr, J. J., Schuchert , B. L., Grenning, J. W., et al. (2009). *Clean code A handbook of agile software craftsmanship*. United States: Pearson Education, Inc.

M.Abbing. (12 April, 2006). Process-data diagram of Open source software development.

McKnight, L. and Fitton, D.. 2010. Touch-screen technology for children: giving the right instructions and getting the right responses. In *Proceedings of the 9th International Conference on Interaction Design and Children* (IDC '10). ACM, New York, NY, USA, 238-241. DOI=10.1145/1810543.1810580 http://doi.acm.org/10.1145/1810543.1810580

Mohs, C., Hurtienne, J., Israel, J. H., Naumann, A., Kindsmüller, M. C., Meyer, H. A., et al. (2006). IUUI–intuitive use of user interfaces. *Usability Professionals, 6*, 130-133.

Open source software development Retrieved 11/16/2011, 2011, from http://chinese-school.netfirms.com/computer-article-open-source.html

OSI Community. (1998). *The Open Source Definition*. Retrieved 26 October, 2011, from Open Source Initiative: http://opensource.org/docs/osd

Perens, B. (1999). The open source definition. *Open Sources: Voices from the Open Source Revolution, 171*, 188.

Pfleeger, S. L., & Atlee, J. M. (2010). *Software engineering, theory and practice* (4th ed.). New Jersey: Pearson Higher Education.

Rahman, M., Gustafson, S., Irani, P., Subramanian, S. Tilt Techniques: Investigating the Dexterity of Wrist Based Input. *CHI'09*

Raymond, 2001 Raymond, E.S., 2001. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, revised ed. O'Reilly.

Ruiz, J. , Yang Li, and Lank, E.. 2011. User-defined motion gestures for mobile interaction. In *Proceedings of the 2011 annual conference on Human factors in computing systems* (CHI '11). ACM, New York, NY, USA, 197-206. DOI=10.1145/1978942.1978971 http://doi.acm.org/10.1145/1978942.1978971

Taylor, B., Bove Jr., V. Graspables: Grasp-Recognition as a User Interface. *CHI'09*.

Utting, I., Cooper, S., Kölling, M., Maloney, J., & Resnick, M. (2010). Alice, greenfoot, and scratch--a discussion. *ACM Transactions on Computing Education (TOCE), 10*(4), 17.

Xiajian, C., Danli, W., & Hongan, W. (2011). Design and implementation of a graphical programming tool for children. *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on, , 4.* pp. 572-576.

Zhou ZY, Cheok AD, Tedjokusumo H, Orner GS (2008) wIzQubesTM-A novel tangible interface for interactive storytelling in mixed reality. Int J Virtual Real 7(4):9-15.

APPENDICES

Appendix 1

Rules of Clean Coding for Code Quality Assurance:

- Refactor & clean up the code for readability and understandable by itself
- Use no abbreviations
- Use as much as possible no comments
- Use good, pertinent names of variables, methods , objects & etc
- The code should be crystal-clear, self-explaining, and we should be annoyingly thorough in making this sure
- Eliminate duplicate code
- Leave no compiler warnings
- Cleanly use known design patterns
- All of above not only for the main code but also for the test code
- Add unit-, regression-, and functional-tests so that the code and functionality is 100% covered.
- Have all the code (include test code) reviewed by independent team members

The Criteria of Open Source Definition (OSI Community, 1998) :

1. Free Redistribution – Copies of the software can be made at no cost.
2. Source Code – The source code must be distributed with the original work, as well as all derived works.
3. Derived Works – Modifications are allowed, however it is not required that the derived work be subject to the same license terms as the original work.
4. Integrity of the Author's Source Code – Modifications to the original work may be restricted only if the distribution of patches is allowed. Derived works may be required to carry a different name or version number from the original software.
5. No Discrimination Against Persons or Groups – Discrimination against any person or groups is not allowed.

6. No Discrimination Against Fields of Endeavour – Restriction preventing use of the software by a certain business or area of research are not allowed.
7. Distribution of License – Any terms should apply automatically without written authorization.
8. License Must Not Be Specific to a Product - Rights attached to a program must not depend on that program being part of a specific software distribution.
9. License Must Not Restrict Other Software - The license must not place restrictions on other software that is distributed along with the licensed software.
10. License Must Be Technology-Neutral – No provision of the license may be predicated on any individual technology or style or interface.

Appendix 2

Gantt chart of Project Tracking

Appendix 3

GNU AFFERO GENERAL PUBLIC LICENSE

Version 3, 19 November 2007

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU Affero General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>.

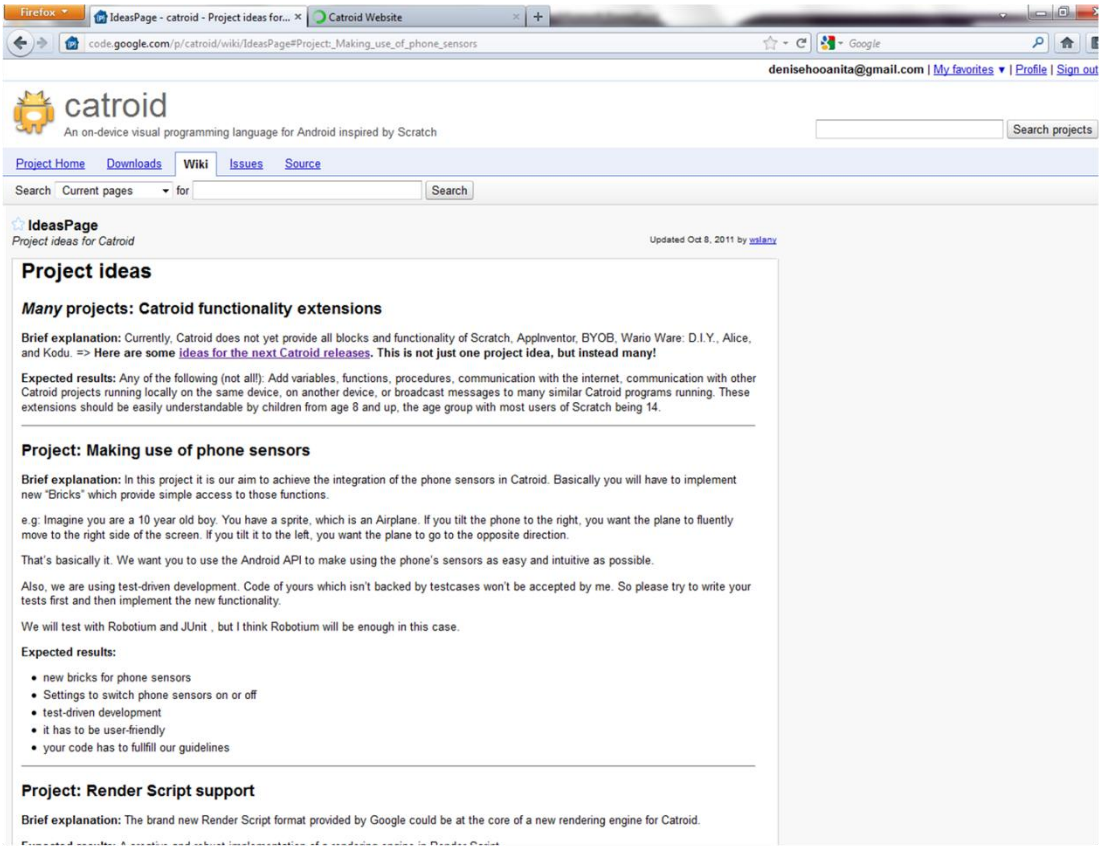
Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see <http://www.gnu.org/licenses/>.

Appendix 4

Wiki Page of Control API for Catroid



Firefox | IdeasPage - catroid - Project ideas for... | Catroid Website

code.google.com/p/catroid/wiki/IdeasPage#Project:_Making_use_of_phone_sensors

denisehooanita@gmail.com | My favorites ▾ | Profile | Sign out

catroid

An on-device visual programming language for Android inspired by Scratch

Search projects

Project Home | Downloads | Wiki | Issues | Source

Search Current pages for Search

IdeasPage

Project ideas for Catroid

Updated Oct 8, 2011 by wslany

Project ideas

Many projects: Catroid functionality extensions

Brief explanation: Currently, Catroid does not yet provide all blocks and functionality of Scratch, AppInventor, BYOB, Wario Ware: D.I.Y., Alice, and Kodu. => **Here are some ideas for the next Catroid releases. This is not just one project idea, but instead many!**

Expected results: Any of the following (not all!): Add variables, functions, procedures, communication with the internet, communication with other Catroid projects running locally on the same device, on another device, or broadcast messages to many similar Catroid programs running. These extensions should be easily understandable by children from age 8 and up, the age group with most users of Scratch being 14.

Project: Making use of phone sensors

Brief explanation: In this project it is our aim to achieve the integration of the phone sensors in Catroid. Basically you will have to implement new "Bricks" which provide simple access to those functions.

e.g: Imagine you are a 10 year old boy. You have a sprite, which is an Airplane. If you tilt the phone to the right, you want the plane to fluently move to the right side of the screen. If you tilt it to the left, you want the plane to go to the opposite direction.

That's basically it. We want you to use the Android API to make using the phone's sensors as easy and intuitive as possible.

Also, we are using test-driven development. Code of yours which isn't backed by testcases won't be accepted by me. So please try to write your tests first and then implement the new functionality.

We will test with Robotium and JUnit , but I think Robotium will be enough in this case.

Expected results:

- new bricks for phone sensors
- Settings to switch phone sensors on or off
- test-driven development
- it has to be user-friendly
- your code has to fullfill our guidelines

Project: Render Script support

Brief explanation: The brand new Render Script format provided by Google could be at the core of a new rendering engine for Catroid.