Vrije Universiteit Brussel

Faculty of Science
Department of Computer Science
and Applied Computer Science

TIAMAT: A Multi-touch Tablet IDE for AmbientTalk

Ayrton Vercruysse

Promotor: Prof. Dr. Wolfgang De Meuter
Advisors: Dries Harnie
Lode Hoste

2012-2013

Contents

Chapter 1

Introduction

1.1 Goal

1.2 Scope

The scope of this project is a third bachelor thesis at the Vrije Universiteit Brussel. The 3th bachelor thesis is given at third bachelor students computer sciences. The goal of this bachelor thesis is to let the bachelor students participate in some actual research at the university. This has been chosen over giving the students a predefined task.

1.3 Definitions and Acronyms

VUB	Vrije Universiteit Brussel
IDE	Integreted Develepment Enviorment
AST	Abstract Syntax Tree
MANET	Mobile Ad-Hoc Network
OHA	Open Handset Alliance
App	Application. Programs on smartphones and tablets
MT4j	Multi-Touch for Java

Chapter 2

Implementation

2.1 Used Software

2.1.1 AmbientTalk

AmbientTalk is a programming language developed at the VUB in 2005. The goal of the language was to focus on making programs within Ad-Hoc networks. This means that AmbientTalk is developed mainly to create programs on mobile devices. AmbientTalk combines elements from other programming languages such like Scheme (closures), Smalltalk (pure OO), Self (prototypes and delegation) and some other languages.

AmbientTalk was originally a part of a doctorate study made by Jessie Dedecker. His goal was to create an extension to the already existing programming language Pic% (pic-oh-oh). Pic% itself was an extension of Pico. Pico was developed by professor Theo D'Hondt in 1996. Pico was developed merely to be used with an educational purposes. Pico is being used as a programming language to illustrate how programming languages are being made.

Pico got it's design principles and concepts from Scheme. The goal of Pico was to be a programming language based on simple rules, easy to extend. Thanks to the simplicity and extendability of Pico this language is often used to do some research to possible extensions and features for programming languages. This is the reason why a lot of offsprings of Pico are created, all with their own special attention to certain problems or extensions. When AmbientTalk was being developed special attention was given create distributed programs within ad-hoc networks. Momentarily AmbientTalk/2 is being used. It's the successor of the original AmbientTalk. Even though AmbientTalk/1 was already successful concerning the programming features for mobile Ad-Hoc networks it lacked some important features to create bigger software applications, such like exception handling,...

In 2006 Tom Van Cutsem and Stijn Mostinckx started developing AmbientTalk/2, the current version of AmbientTalk. The changes between AbmientTalk/1 and AbmientTalk/2 are quite big. There has been made an entire new design for most aspects of the language, including the syntax. The reason to change the syntax was to make AmbientTalk more accessible for people who don't have any experience with Pico.

AmbientTalk is still mainly used by students at the VUB to do some research. But, slowly but surely, AmbientTalk became a useful, handy programming language which was usable to create some rather big programs. Because the focus of AmbientTalk lies within the distributed networks and this is still an area in which a lot of research is being done there aren't a lot of good alternatives. This made AmbientTalk a good alternative to create some real-life software.

The symbiosis with the underlying Java Virtual Machine enables the possibility to use some parts of the Java programming languages within AmbientTalk, which makes it possible to combine both languages within one project.

The renewing element of AmbientTalk is so big that it became the main subject of the Distributed and Mobile Programming Paradigms course taught at the VUB.

2.1.2 Google Android

General

Android is an open source platform for mobile devices, owned by Google. Android is developed by Android Inc., a company bought by Google in 2005, which later placed it within the Open Handset Alliance (OHA).

It's an on Linux based operating system and is mainly programmed in C (core), Java (UI) and C++.

Android is originally created by Android Inc., a company founded by Andy Rubin, Rich Miner, Nick Sears and Chris White in 2003. Their goal was to create an operating system for " ... smarter mobile devices that are more aware of its owner's location and preferences."

On October 17th 2005 Google bought Android Inc. and made it a part

Figure 2.1: Android logo

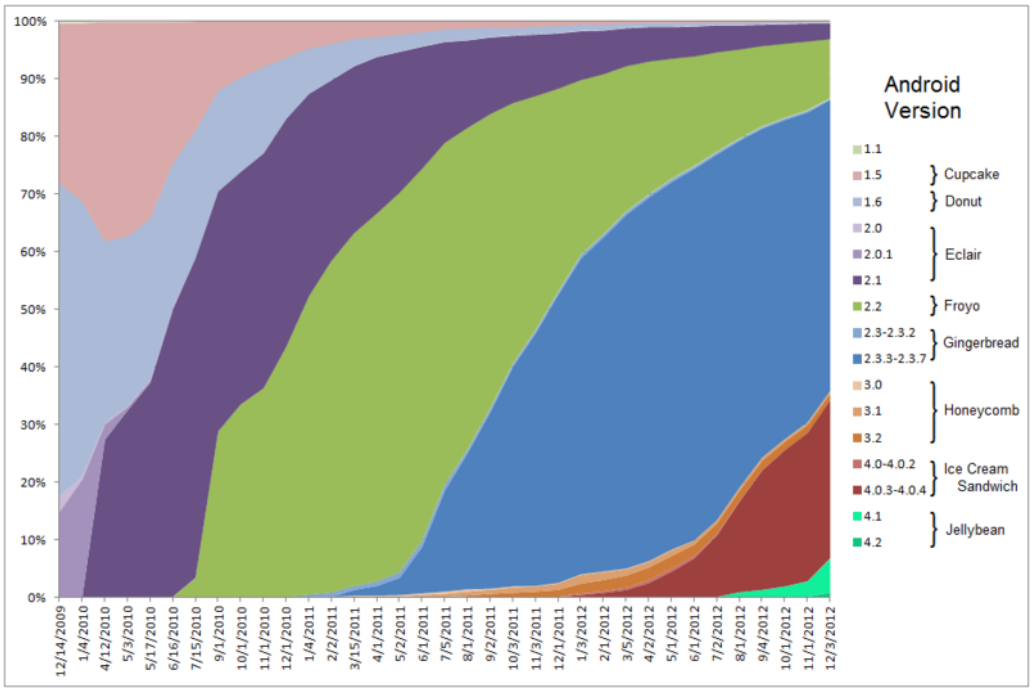

Figure 2.2: Android historical version distribution

of their company. Even after Google bought the company founders Andy Rubin, Rich Miner and Chris White kept working for Android. Already back then the goal of Google was to focus on the market of mobile devices with Android.

Only on December 9th 2009, together with the founding of OHA, the first product of Google concerning the market of the mobile devices was announced; Android. An on Linux kernel version 2.6 based platform for mobile devices.

According to estimations in the second quarter of 2009 approximately 2.8% of new smartphones had Android as operating system. In the fourth quarter of 2010 this number had grown to 33% what made it the best selling smartphone platform. In the third quarter of 2011 this would already have grown to 52.5%. In February 2012 Andy Rubin said that Google activates 850000 smartphones or tablets on a daily basis.

Versions

Through the years multiple versions of Android have been released. In November 2007 a beta version was released, but only on September 23th 2008 Android 1.0 was released. Android 1.0 was introduced on the HTC Dream. This version was already equipped with the Android Market, an application which gives you the option to download and update new applications, camera support, synchronization

with, and use of, most Google products, like Gmail, Google Calendar, Google Contacts, Google Search, Google Talk,... Also Wi-Fi and bluetooth were already supported, just like a Youtube media player.

After Android 1.0 in February 2009 Android 1.1 was released. After Android 1.1 a next version of Android, Android 1.5 which was given the name Cupcake was released followed by Android 1.6, called Donut. In October 2009 Android 2.0 got released. It was called Eclair. Eclair was followed by Froyo (2.2.x) and later the popular Gingerbread (2.3.x) Next to the 2.x series in February 2011 the 3.x (Honeycomb) version got released. This version was created with focus on tablets and no longer on smartphones.

In October 2011 Android 4.0.1 (Ice Cream Sandwich) was presented. ICS was the version of Android which should combine the 2.x series and 3.x series to the 4.x series. This means that there will be no longer 2 parallel versions of Android, but only one, working on both smartphones and tablets. This made it easier for Google, so they didn't need to keep two versions up to date but only one. On July 9th 2012 a new version of the Android 4.x series was released named Jelly Bean (4.1). In Figure 1.2 one can see a chart showing global Android version distribution from November 2009 to December 2012.

Design

Android is based on a Linux kernel, with middleware, libraries and APIs written in C and application software in Java. In Figure 1.3 the architecture of Android is shown. The top layer of the figure; the Application Layer are the applications which are deployed on the phone when bought, like an SMS program, calendar, internet browser,... All these applications are written in Java. The application framework is made to create an open developing platform. Android gives developers the possibility to develop rich and innovative applications, by giving them the liberty to make use of all hardware, like location information, setting alarms, add notifications to the status bar,...

The libraries layer contains a number of C/C++ libraries which are being used in multiple layers of the Android system. Developers are able to explore these libraries trough the Android Application framework for developers. Some of these libraries are SGL (an underlaying 2D engine), FreeType (a bitmap and vector renderer), MQLite (a database engine),... The Android Runtime part is a number of core libraries which gives most Java functionalities. Last there is the Linux kernel which provides safety, memory management, process management, network stacking and driver models. It also functions as abstraction layer between the hardware and the rest of the software.

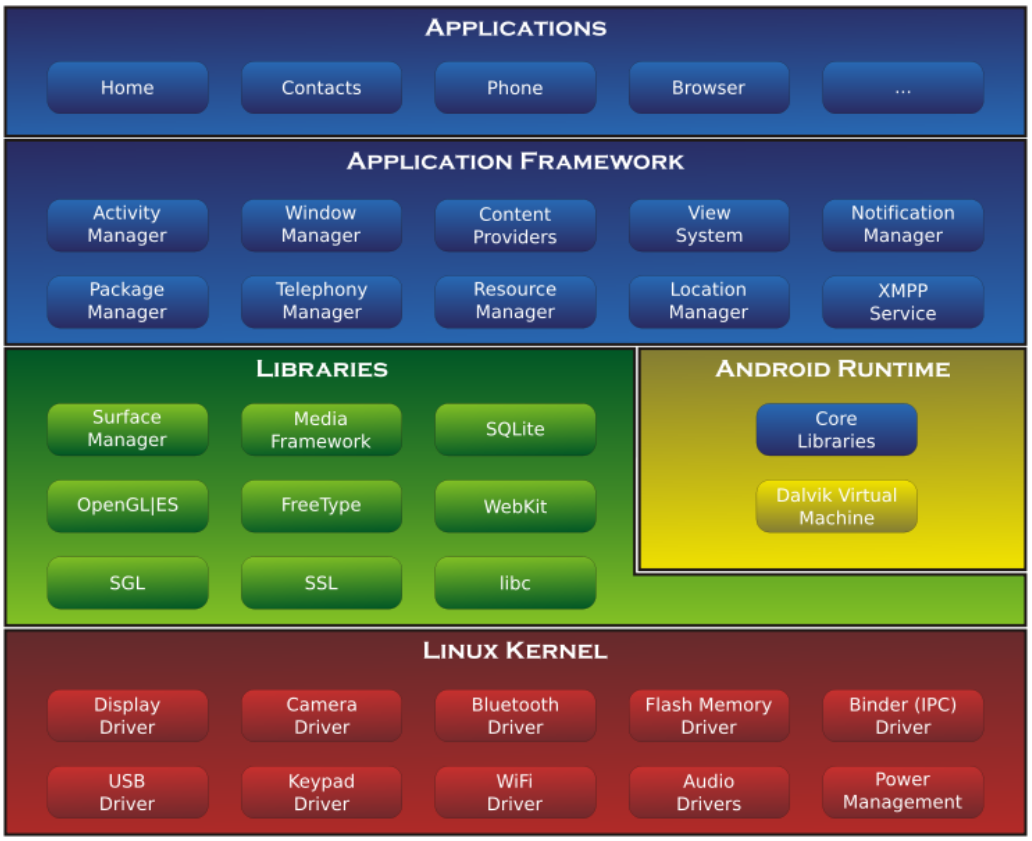

Figure 2.3: Android System Architecture

Open Handset Alliance

The Open Handset Alliance (OHA) is a consortium of 84 firms to develop open standards for mobile devices. Practically this translates in promoting Android. OHA is founded on November 5th 2007, with as main founding company Google, together with 34 other firms. These firms are active in creating software, creating mobile devices or creating chips. The main goal was to make Android a worthy competitor of other mobile platforms like those developed by Apple, Microsoft, Nokia (Symbian), HP, Research in Motion and Barda.

Some of the companies who aided founding OHA were, besides Google, HTC, Sony, Dell, Motorola, Qualcomm, Texas Instruments, Samsung Electronics, LG Electronics, T-Mobile, Nvidea,...

2.1.3 MT4j

General

Multi-Touch for Java is developed to make it possible to create multi touch applications within the Java programming language. It's an open source Java Framework. It offers a GUI in which we can use shapes like rectangles, rounded rectangles, ellipses, polygons,... To each of these forms gestures can be attached. These can be in 2D or 3D. Next to the predefined shapes you can also make use of buttons, text areas, sliders and a multi touch keyboard.

Architecture

The architecture of MT4j is built in multiple layers which communicate with each other. Using event messages which are sent from one layer to another. The different layers within MT4j are the Presentation Layer, Input Processing Layer, Input Hardware Abstraction Layer and the Input Hardware Layer.

Input Hardware Layer

By making use of the Input Hardware Abstraction Layer MT4j can give support to different kinds of input hardware, while only little alternations are needed within the Hardware Abstraction Layer. Within this layer all raw input gets reformed to unified input events. The only change needed when one wants to use a new form of input is to change the abstract superclass of all input sources and the specific input for this new part of hardware. MT4j already has a number of input methods which are supported, like a keyboard, mouse,... All input methods can be used synchronously and together without chances on conflicts.

Input Processing Layer

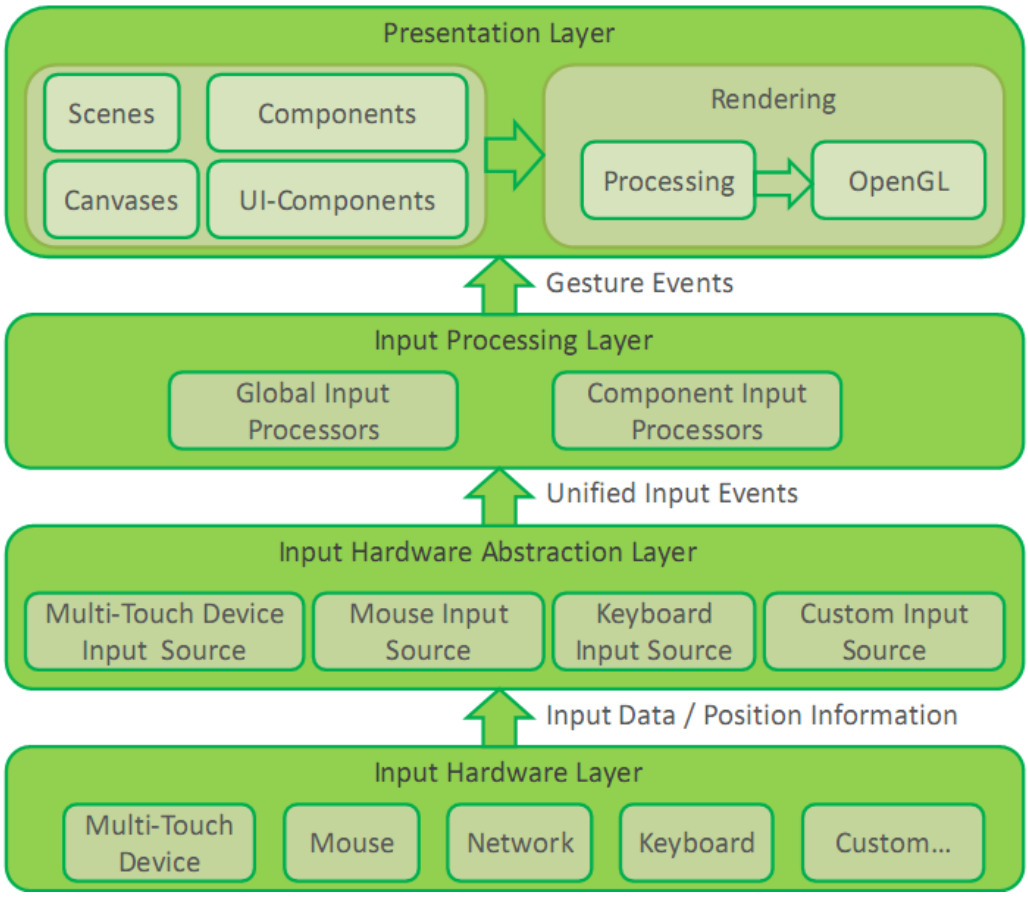

Figure 2.4: MT4j Architecture

In MT4j the input processor gets used on two moments within the input event flow. The first moment is when input directly from the Input Hardware Abstraction Layers comes, the other moment is at the component level. This means one can recognize gestures which are meant for only one component.

Presentation Layer

The Presentation Layer within MT4j uses scenes. Scenes are made to be able to separate different aspects of a program, input and output, in a nice clean encapsulated manner. An example of multiple scenes could be a game where there's one scene to create the menu and an other scene in which the actual game is played.

Next to scenes MT4j also uses components. Components are parts of a program which can be connected in a hierarchical manner. So one can use a basic component and add multiple components within this component. Here we can see as the basic component the background of the application, and within this component one or more other components are used.

The most basic component of MT4j is a canvas. A canvas is the link between multiple components and the input of the hardware. It can check what other components are on what place within the screen at a given time.

To render the MT4j components they use the Processing toolkit. Processing is an open source toolkit that is used to create data visualization and interaction.

MT4Android

MT4Android is an Android version of MT4j. Because MT4j lays the focus on multi touch and is made within Java it seemed a logical step to port MT4j to the Google Android platform. MT4Android is made by a coworker of MT4j. MT4Android still isn't a full version, but there is a beta version available.

Chapter 3

Implementation Details

3.1 ASTs

3.1.1 Making ASTs

Description: Every piece of code used in the IDE will be directly linked to an AST. For this matter there has to be a possibility to create new ASTs from scratch. To create new ASTs we will need to implement the different parts of which an AST can exist. These different parts of an AST will be called a Nodes. We will start with a list of possible nodes with which basic programs can be made. Later extra kinds of nodes can be added to the AST. A superclass Node will be implemented which will organize the tree structure by providing operations like getParent(), which gets the parent of a Node, setChild(oldChild, newChild) which will set a child of a node and getChilderen() to get the children of a Node. By using these operations new trees can be created. The types of nodes that will be implemented are: Placeholder, Begin, Block, Definition, Value, Argument List, Function, Function Call, Function Definition, Operation, Table, Table Call Table Definition.
Priority: High
Status: Within the vub.ast package the implementation of the Nodes can be found.

3.1.2 Deleting ASTs

Description: When we have an excising AST and parts of this AST became irrelevant there should be a possibility to remove this AST. The removing of an AST, in total, or only a part of an AST will happen by replacing this AST by the Placeholder node. This possibility will be, as said in 1.1 implemented in the Node

class, by means of the setChild(oldChild, newChild) function.
Priority: High
Status: The possibility to set children of nodes of implemented in each node. This means that we can delete nodes by setting the node we want to delete to null in the parent node.

3.1.3 Replacing parts of ASTs

Description: When new Nodes will be created it's children, if there are any, will be made by Placeholders. When making a function for example, we'll implement the operation, but the two operands will be initialized with a Placeholder node. To be able to change these Placeholders to more relevant Nodes we should be able to replace existing Nodes by other Nodes, in this example maybe a Value, or an other Operation. This can be done on a similar way as the deletion of ASTs, by making use of the replaceChild(oldChild, newChild) method.
Priority: High
Status: In the same way of deleting ASTs replacing parts of an AST can be done. We just set the child we want to change to the new node. These possibilities are implemented the Node class.

3.1.4 Rendering ASTs to text

Description: To be able to pass the constructed code to the AmbientTalk interpreter we have to convert the AST to an understandable format for the interpreter. The interpreter works with a text file containing AmbientTalk code. This means that the ASTs have to be converted to text. Converting an AST to a text should be a function within each Node, calling the same function recursively on all of it's children.
Priority: High
Status: Every node can be rendered to text. There's a function foreseen in every subclass of the Node class, called toString(). Each of these call the toString() function on it's children as well.

3.2 Templates

3.2.1 Templates creating ASTs

Description: For convenience we will make use of templates. Templates will be frequently used parts of code that will be saved. The way templates will be stored in the memory will be by an AST. Every time a template is used withing the program a copy of this AST will be made and inserted into our current program. These templates can be frequently used functions, but mainly these will be the ASTs representing basic elements of the language like the if-then-else structure, when-discovered,...
Priority: High
Status: There is a file called 'templates.xml' containing all templates that we currently use. Within the TemplateReader class these templates are being read and translated to ASTs.

3.2.2 Creating templates from XML

Description: To keep a clear view on what template functions we have, and make it easy for users to add templates, without having to know the implementation of the entire program we will keep the Templates in a separate XML file. When having an XML file containing the structure to create new Templates a function will be implemented that creates, from the XML file, a new AST, which can be used as a Template within the program.
Priority: High
Status: There is a file called 'templates.xml' containing all templates that we currently use. Within the TemplateReader class these templates are being read and translated to ASTs.

3.2.3 Create templates from functions

Description: When a newly created function is often used by the user and he wants to save this function, the possibility to write this function, in correct XML, to the existing XML file will be provided. This means that whenever the program is terminated functions can be stored inside the XML file.
Priority: Low
Status: Every new function that is being created is saved as a template within the program. As every node has the possibility to be translated to XML it is fairly

easy to add the option to write this XML to an XML file.

3.3 Interface

3.3.1 Rendering of nodes

Description: Each type of node has to have a screen representation. For each node we need a function that displays the Node on the screen, and that recursively calls the display function of all children of this node. We won't implement these displays within the separate types of nodes, discribed in 1.1, but for each node we will implement a separate display function. This with as goal that we can re-use our AST implementation without having to dead with the program specific display possibilities.
Priority: High
Status: Within the vub.ast.Renndering package all needs to render a node to the screen is kept.

3.3.2 Lay-out engine

Description: We will make a render manager, this manager will render all nodes and make sure they are organized on a ordered way. The render manager will use indention to organize nodes across the screen. **Priority:** High
Status: Within the vub.rendering package we have the RenderManager class. This class gives you the possibility to render nodes by sending MTRectangles to them, and tell the class how to render them. This being next to the previous node or under the previous node. The is also to option to indent the next function. This for when one want to show a new node under the previous one, but indented.

3.3.3 Creating functions on the spot

Description: Whenever a new function (or variable) is created within the program a link to call this function will be added to the menu's. This means that we can easily re-use a made variable or have an easy way to call an earlier created function.
Priority: High
Status: Newly created functions will have a automatically have a functionCall

made. This will be added to the list of possible functioncalls.

3.3.4 Buttons

Menu's

Description: To keep track of all templates or calls that can be made we will need a menu that gives us the possibility to insert templates in to the code. We will use one a menu with some sub menu's, divided for each kind of templates. These can be provided functions self implemented function, used variables,...
Priority: High
Status: In the package vub.menus all menus that are availeble in the program are implemented. These are a standard menu, a menu for functions, operations, own functions, definitions, variables, ...

Run button

Description: Thanks to requirements 4.2 we can evaluate made code with an external AmbientTalk app. To do this we'll add a button to the main screen which will first make sure requirement 1.4 is done, the creation of a text file with the current code, and later on run the AmbientTalk app, as described in requirement 4.2. **Priority:** High
Status: The implementation of this button can be found in the Tiamat class within the vub.tiamat package.

Delete button

Description: To be able to delete some code, as described in 1.2 we need a way to select what code will be deleted. This is why we will implement a delete button on the screen, which deletes the current selected piece of code. This can be done by pressing this button or by dragging the selected code to this place. **Priority:** High
Status: The implementation of this button can be found in the Tiamat class within the vub.tiamat package.

3.3.5 Views

Colorcoding keywords

Description: As in many languages the use of color coding should be done here aswell. We will try to achieve the same color coding as being used in the official AmbientTalk IDE in Eclipse.
Priority: Low
Status: The implementation of MT4j does not allow us to make use of color coding. Because of a bug within the code it is impossible to change the color of only 1 word. This is on the other not the problem when one to change the background color of a MTTextArea.

Colorcoding types

Description: Next to the color coding of keywords we will implement color coding on the types of words. This means that we will give Placeholders different colors, just like VariableNames,...
Priority: Low
Status: Unlike the color coding of keywords the background color of MTTextAreas can be edited.

Codefolding

Description: The use of AST should gives us the possibility to easily fold in certain parts (read piece of the AST) of our code. This can be done by just not rendering certain nodes, and it's children, of the AST.
Priority: Low
Status:

Move code

Description: By using the possibility of deleting parts of ASTs and saving parts of ASTs a copy-pasty system, or even moving of code to places that make sense belongs to the possibilities.
Priority: Low
Status:

3.3.6 Gestures

Selection of code

Description: To be able to replace, delete, move,... code we need a possibility to select certain parts of the code. This will be done by a tap or doubletap gesture. The selected code will be marked and it will be possible to makes changes to this code.
Priority: High
Status: Code can be selected by a simple tap on top of it. The selected code will be surrounded by a red square to indicate it has been selected.

Pinch-to-zoom

Description: When starting on a piece of code by using the pinch-to-zoom function the selected part of code should be extended. This can be done by enlarging the selected area from the current AST to the parent of this AST.
Priority: Low
Status: By zooming on a certain piece of code more parts of the code will be selected.

90 degrees turning for comments

Description: When a piece of code is selected, and afterwards this piece of code gets turned around for 90 degrees, this piece of code should be commented out. Commented code should not be written to the text file when this is called on this AST.
Priority: Low
Status: By turning the screen 90 code will be commented out.

Fast scrolldown

Description: When we want to scroll down a longer piece of code we can use the scroll with two fingers gestures to get us scroll faster.
Priority: Low
Status: By scrolling over the code with 2 fingers you can scroll down the code.

3.4 Evaluating code

3.4.1 Writing code to text file

Description: To make use of the AmbientTalk app we need a text file with the AmbientTalk code. We can make use of the toText() function of each node to translate our current AST to plain text, and afterwards we will need to write this text, into a text file.
Priority: High
Status: The function to write the code to a file can be found inside the action of the run button in the Tiamat class. Also the outputstream used for this can be found in this class.

3.4.2 Call external AmbientTalk app

Description: The external AmbientTalk app, which makes use of the text file from the previous - insert number- should be called. To get this app called we will implement a button on the screen, which first gets the text file created and afterwards calls the AmbientTalk app.
Priority: High
Status:

3.4.3 Return to TIAMAt after evaluating code

Description: The AmbientTalk app will evaluate the program. After the evaluation of this program and returning the result we should get back the our application to create the possibility to edit our current program
Priority: High
Status:

3.5 Advanced Interactions

3.5.1 Extra interface for comments

Description: An extra interface can be created in which we can store extra comments. This interface should only be called on object which could get any use of extra comments.

Priority: Low
Status: There is an extra interface for comments implemented, but currently not in use.

3.5.2 Speaking comments

Description: A nice feature is to add spoken comments to our program. The possibility to add spoken comments will be integrated in the extra interface for spoken comments, announced in requirement 5.1. There will be added extra buttons to record comments and play these comments.
Priority: Low
Status:

3.5.3 Selector for Java classes

Description: Within the AmbientTalk language it's possible to make use of a Java class selector. Adding this feature to this implementation will make sure that our project becomes a better alternative to the pc one.
Priority: Low
Status:

3.5.4 Multiple tabs

Description: If we want to make bigger programs it would be easy to be able to have multiple tabs. Certainly when big programs exist of multiple files. To make this possible we would need to save multiple AST trees and implement a possibility to switch between the different tabs.
Priority: Low
Status:

3.5.5 Saving files

Description: As we will implement a write to XML to save new Template into an XML file we could use this procedure to save an entire program in XML, and later rebuild our AST and on this way reconstruct our program.
Priority: Low
Status: Every Node has the possibilty to show it's own XML representation.

This representation can be saved by clicking the save button. Whenever the app starts again there will be asked if you want to open previously stored code.

Chapter 4

Use case

Chapter 5

Reflection and future work

Chapter 6

Bibliography