Introduction to MEE: The Mobile Experience Engine

About this Document

This document provides an introduction to the MEE Engine for all users.

Background to MEE

What is MEE?

The MEE Engine is a software development kit for creating advanced applications and media-rich experiences on mobile devices. It is particularly designed for creating mobile experiences that rely on the user's position in the real world (called '***locative***' mobile experiences) and that rely on sensor input either from user actions, such as gestures, (which we call '***activity***' data) or on data from the environment such as light levels and noise levels (which we call '***environmental***' data).

History

MEE was created by mobile applications researchers in Canada, working on a project entitled 'Mobile Digital Commons Network' or '***MDCN***', funded by Canadian Heritage. The purpose of the project is to explore next generation mobile content and applications, and in particular locative rich-media experiences and multi-player games set in some of Canada's most stunning geographical heritage such as the Banff National Park. To-date, the research has generated a number of applications, including 'Memichi' a locative game set in Banff National Park.

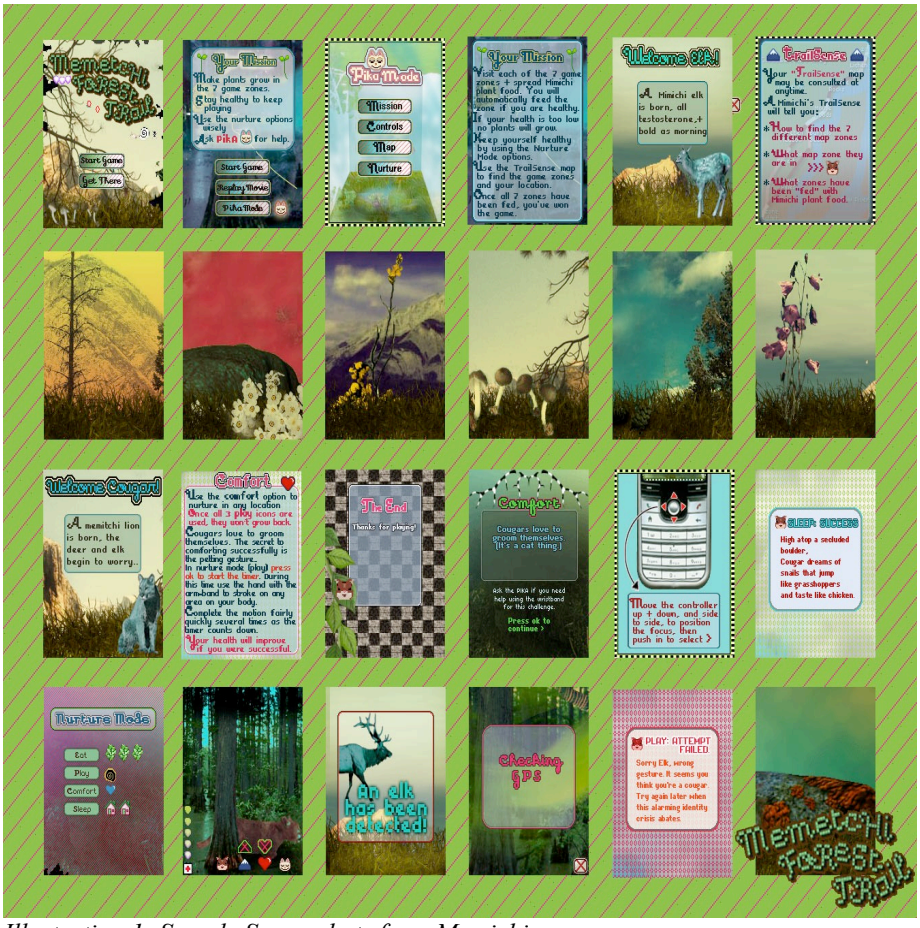


Illustration 1: Sample Screenshots from Memichi

This research explored many aspects of mobile application development that have not been explored previously, including mobile application design for outdoor environments, rich-media design for specific locations, mobile-augmented reality and gestural inputs to mobile applications. As a research project, there was a need for rapid prototyping and testing, while always aiming to produce final products that

could be enjoyed by real end-users.

MEE evolved during the research as the only means by which high-quality experiences could be rapidly prototyped on mobile platforms while also achieving a high quality, stable product.

Why

Mobile device application development differs considerably from more traditional application and experience development. A more detailed review of the challenges and differences is provided in accompanying documentation, but in brief:

- Mobile devices have ***insufficient performance*** and too much variety to support client-side interpretation of high-level languages such as Javascript or XML (typical in web design situations) for any but the simplest of applications;
- Mobile devices have ***widely differing capabilities***, limitations and usage and this means that tools must be re-designed from scratch to deliver meaningful capabilities;
- Locative and sensor-based experience ***design is challenging***, and in particular, ***testing is time-consuming*** and difficult;

Due to these limitations, high-quality mobile experiences need to be developed in lower-level languages such as Java and C++, and targeted to specific devices.

However, for designers to rapidly explore and develop mobile experiences, C++ and Java present significant roadblocks. Few designers are themselves competent lower-level programmers, and are thus less able to 'explore' possibilities themselves, and development in these languages, particularly for cellphones, is extremely time-consuming.

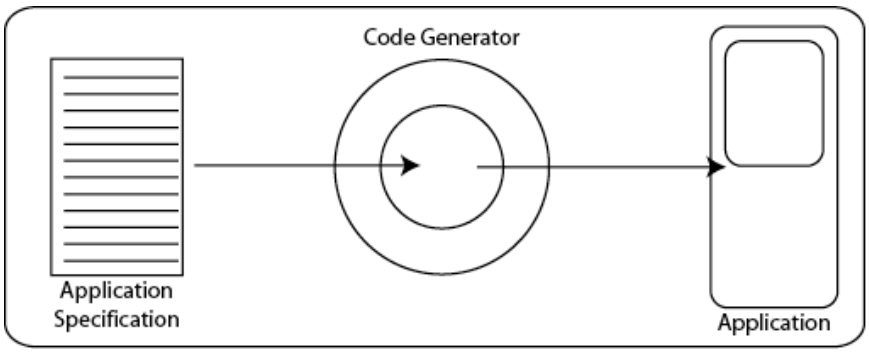
However, given the flexibility and newness of the design domain, designers are not typically in a position to specify exactly an application before developers begin working, and given device constraints, engineers are typically unable to predict exactly what functionality will or won't be possible without developing test implementations.

This Catch-22 typically leads to very slow application development, and expensive budgets with a low probability of application success.

Overview of MEE

MEE is a software development development kit designed to overcome many of the problems associated with developing high-performance rich-media mobile applications. It allows designers to work in simple high-level languages (specifically XML) and automatically generates device-specific C++ and Java code. It enables developers to rapidly add capabilities and incorporate them in the auto-generation mechanism. At its heart, MEE is a XML to C++ (or Java) code-generator-generator tailored for locative and sensor-based mobile applications.

For Designers



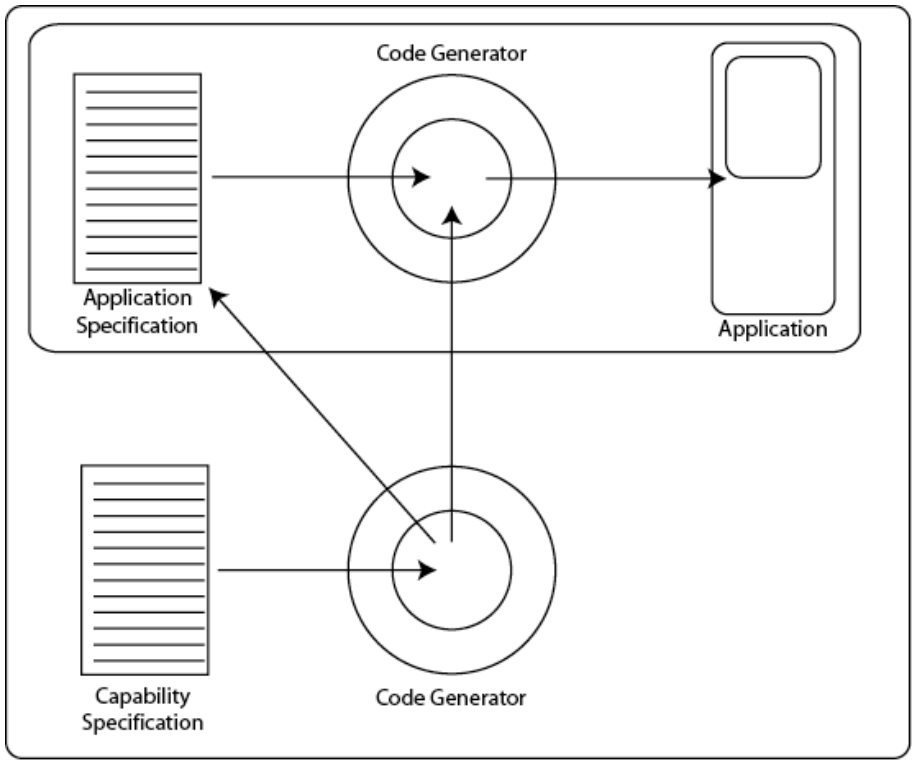
Designers (or application developers) working with MEE work in XML, a simple high-level language for which many powerful editing tools exist. Once their XML code is written, MEE processes it and automatically generates and builds complete high-performance C++ or Java code specifically for each device on which the application will run. Because application code is automatically generated, common bugs are eliminated and the resulting code is of a high quality. Because low-level code is created specifically for each device, MEE achieves high performance.

For Developers

Developers working with MEE are not required to perform application development, other than assisting with writing XML descriptions of the application. Developers can therefore focus on creating high quality libraries of capabilities for use by the auto-generation mechanism.

Unlike most code-generators, MEE incorporates new libraries of capabilities into the code-generator, without requiring developers to re-code the auto-generation mechanism, which can be a time-consuming effort that can introduce new errors.

MEE provides a code-generator-generator. That is, the code-generator that the Designers use is actually generated by MEE and not by the project software engineers.



Once a developer has a clear idea of the software library of capabilities they wish to include in the MEE system, they create an XML specification in a format defined by MEE. This specification details classes, methods, variables etc that the library will make available to the auto-generation mechanism.

From this XML specification of a software library, MEE performs two key tasks:

- It ***generates library template*** files containing most of the code to implement the library, to reduce the effort for the developer in matching their code to the specification;
- It ***generates an XSL code-generator*** used to transform application designs into working C++ and Java applications.

Because MEE handles updating the code-generator to incorporate new library functionality, and produces template library classes for each library, it radically reduces the amount of effort required to implement new libraries of capabilities and enable auto-generation of applications against them, while improving the quality of the code.

Other Capabilities

Because both applications and software libraries are specified through XML, MEE is able to provide additional tools to assist in the software development process, including:

- Allowing designers to write in a 'Simple XML', a version of the full application design XML cut-down for their specific application domain, to further simplify application design;
- Transform the application specification into SVG and from there into images or 'blueprints' that enable detailed visualisation of the application functionality in a variety of forms more easily analysed than a text XML document. Because these visualisations are produced from exactly the same file that produces the final application code, they enable accurate analysis of application logic and activity;
- Generate tests that stress library and application features to ensure that the operate correctly even under more extreme circumstances that might be expected in user situations, and hence improve the quality of the code.

How Does MEE Work

Please see MEE_Technical_Intro for more details.

MEE is based on XML technologies (including XML, XSD, and XSLT), as well as Java or C++ as specific to devices being targeted.

To use MEE, an application developer writes an ***Application.xml*** file that describes the application they wish to build. From a application developer's perspective, MEE performs the following tasks:

- Validates an ***Application.xml*** document against a ***Specification.xsd*** schema to ensure that the application can be built against the following libraries;
- Transforms an ***Application.xsd*** document into a single application code file (e.g. .cpp in C++) that implements the application design
- Builds the application code file against the libraries to produce a final executable

In addition, MEE can

- Transform the ***Application.xsd*** document into a number of SVG (scalable vector graphics) documents which can then be rasterised or opened in a suitable viewer (e.g. Adobe

illustrator) to review the design in a visual manner
- Create an installer file for each device to simplify the process of downloading the file to a particular device for testing.

To use MEE, a software developer writes an ***Libraries.xml*** file that describes libraries they wish to create, and implements those libraries in C++ or Java, usually within the templates provided. From a software developer's perspective, MEE performs the following tasks:

- Validates the ***Libraries.xml*** file against the core ***MEE_Library_Schema.xsd*** schema file to ensure that it is a valid library definition file;
- Generates templates for all the libraries and classes defined in the ***Libraries.xml*** file, which the developer may then implement;
- Generates a ***Specification.xsd*** schema that validates that ***Application.xml*** file can build against these libraries;
- Generates a ***ApplicationGenerator.xslt*** document that is able to transform an ***Application.xml*** file that validates against the generated ***Specification.xsd*** into C++ or Java code that can be successfully built and executed against the libraries.

The exported definitions of libraries consist of subclasses of a MEE core library (implemented in both Java or C++) that define how different elements of an application connect together. Generated application classes subclass library classes to provide final implementation.

Third-party Tools

MEE relies on third-party open-source, easily available software tools to perform xslt transformations and run build and generation tasks. In particular, the build mechanism uses Ant while the xslt transformations are performed by default by xalan. MEE thus integrates easily with common open-source integrated development environments such as NetBeans or Eclipse.