Embodiments of the invention provide a platform-independent application development framework for programming an application. The framework comprises a content interface configured to provide an Application Programming Interface (API) to program the application comprising a programming code to be executed on one or more platforms. The API provided by the framework is independent of the one or more platforms. The framework further comprises an application environment configured to provide an infrastructure that is independent of the one or more platforms and one or more plug-in interfaces configured to provide an interface between the application environment and the one or more platforms.
FIG. 1
Utility Module 406

Control infrastructure Module 408

Asset management Module 410

Plug-in Interface 412

Plug-ins 414

FIG. 4
FIG. 7

Abstract Functionality

Connect-server 706
Read bytes 708
Disconnect-server 710

Plug-in Interface

Platform Specific Functionality

System socket is created and a connection attempt is made

Bytes are read from the socket

System socket disconnects and released
Start

Provide an Application Programming Interface (API) to program an application comprising a programming code to be executed on one or more platforms 802

Provide an application environment to program the application, wherein the infrastructure is independent of the one or more platforms 804

Provide an interface between the application environment and the one or more platforms 806

STOP 808

FIG. 8
PLATFORM-INDEPENDENT APPLICATION DEVELOPMENT FRAMEWORK

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application draws priority from U.S. Provisional Application No. 61/152,211, filed on Jun. 16, 2008, and hereby incorporated by reference herein in its entirety.

FIELD OF THE INVENTION

[0002] The teaching herein generally relates to application development framework and more specifically to platform independent development framework for applications.

BACKGROUND OF THE INVENTION

[0003] Various users across the globe communicate or perform various activities on computer and device networks. Moreover, the users interact with each other through the networks, such as the Internet. Typically, devices like personal computers are used by the users to interact over the Internet. The users can interact from various Internet websites or social networking sites, for example, Facebook, Myspace, Hi5, and Orkut etc. Recently, the development in mobile devices such as cell phones, smartphones and PDAs, computers, laptops and the like has enabled them to be used for performing various activities on networks such as the Internet. Moreover, the mobile devices can be used for real-time interaction with other users on the network. The interaction or communication can be in the form of chatting, playing interactive online games, browsing, shopping, music, videos, banking, business and the like.

[0004] The rapid pace of innovation in technology has generated various types of devices and platforms. Moreover, the number of devices is increasing rapidly. For example, there are various operating systems available for the devices such as Windows, Linux, Macintosh, and Symbian, etc. Moreover, a large number of J2ME platforms are available for the mobile devices such as cell phones. Furthermore, the mobile devices have a wide range of capabilities in terms of screen size, screen type, screen resolution, processor, and memory etc. The applications for these devices have to be developed based on their platforms. As a result, each application has to be developed based on their platform or ported to other platforms. For example, in case of computer games the programming language when not supported by a device platform is translated manually from one programming language to another. However, manual translation requires specific and in-depth knowledge of the programming languages of the different operating systems. Moreover, manual translation is a very time consuming process. Furthermore, the programming languages are constantly developed or get modified. Moreover, the applications have to be developed on specific development platforms compatible with the programming language used for development and the operating system of a device. Therefore, knowledge of specific tools and Application Programming Interface (API) for every device platform may be required to program an application. Some existing mechanisms, such as emulation of applications and virtual machines are available to support applications on multiple devices.

[0006] In case of virtual machine mechanisms the programming code of the application is separated into platform independent code and platform dependent code. Platform independent code is written in the virtual machine's coding language, which usually is a scripting language. Further, the code may be compiled to a binary that only the virtual machine itself knows how to execute. Since the code is never compiled to a native platform binary and must run inside the virtual machine space, there exist runtime overheads. Moreover, the existing mechanisms generate application program codes that are large or require heavy processing time. The problem is compounded with the continuously increasing number of device platforms and technology.

[0007] What is needed in the art is a framework that is required for development of applications independent of the platform of the device.

SUMMARY

[0008] The invention provides a platform-independent application development framework. The framework comprising: a content interface configured to provide an Application Programming Interface (API) to program an application, the application comprising a programming code to be executed on one or more platforms, wherein the API is independent of the one or more platforms; and one or more plug-in interfaces configured to provide an interface between the application environment and the one or more platforms.

[0009] The invention further provides a platform-independent application development framework. The framework comprising: a content interface configured to provide an Application Programming Interface (API) to program an application, the application comprising a programming code to be executed on one or more mobile device platforms, wherein the API is independent of the one or more mobile device platforms; an application environment configured to provide an infrastructure to program the application, wherein the infrastructure is independent of the one or more mobile device platforms; and one or more plug-in interfaces configured to provide an interface between the application environment and the one or more mobile device platforms.

[0010] The invention further provides an architecture for platform-independent development of an application, the application comprising a programming code to be executed on one or more mobile device platforms, the architecture comprising: an abstraction layer configured to: provide an Application Programming Interface (API) to program the application, wherein the API is independent of the one or more mobile device platforms; provide an infrastructure to program the application, wherein the infrastructure is independent of the one or more mobile device platforms; and provide an interface between the abstraction layer and the one or more mobile device platforms.

[0011] The invention further provides a method for platform independent development of an application. The method comprising: providing an Application Programming Interface (API) to program an application, the application comprising a programming code to be executed on one or more platforms, wherein the API is independent of the one or
more platforms; providing an infrastructure to program the application, wherein the infrastructure is independent of the one or more platforms; and providing an interface between the application environment and the one or more platforms.

[0012] An article of manufacture for platform independent development of an application, comprising: an electronically accessible medium including instructions, that when executed by a processor, cause the processor to: provide an Application Programming Interface (API) to program an application, the application comprising a programming code to be executed on one or more platforms, wherein the API is independent of the one or more platforms; provide an infrastructure to program the application, wherein the infrastructure is independent of the one or more platforms; and provide an interface between the application environment and the one or more platforms.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] Having thus described the invention in general terms, reference will now be made to the accompanying drawings, which are not necessarily drawn to scale, and wherein:

[0014] FIG. 1 is a simplified block diagram of a platform independent development framework;

[0015] FIG. 2 shows functional overview of abstraction layer according to an embodiment of the invention;

[0016] FIG. 3 shows functional overview of pre-processor layer according to an embodiment of the invention;

[0017] FIG. 4 shows functional overview of a platform independent development framework according to an embodiment of the invention;

[0018] FIG. 5 shows functional overview of a control infrastructure module according to an embodiment of the invention;

[0019] FIG. 6 shows functional overview of a asset management module according to an embodiment of the invention; and

[0020] FIG. 7 shows functional overview of a plug-in interface according to an embodiment of the invention.

[0021] FIG. 8 is a flowchart illustrating the method for platform independent development of an application, according to an embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0022] Illustrative embodiments of the technology now will be described more fully hereinafter with reference to the accompanying drawings, in which some, but not all embodiments of the technology are shown. Indeed, the technology may be embodied in many different forms and should not be construed as limited to the embodiments set forth herein; rather, these embodiments are provided so that this disclosure will satisfy applicable legal requirements. Like numbers refer to like elements throughout.

[0023] As a preliminary matter before exploring details of various implementations, reference is made to FIG. 1 for illustrating a simplified block diagram of a platform independent development framework. FIG. 1 discloses a platform independent framework that allows a rapid platform independent application development, without the need for an application developer to know each platform. As shown in FIG. 1, devices 102a, 102b, 102c and 102n are connected through an application development framework 104, which in turn is connected to application contents 106a, 106b, 106c and 106n. Devices 102a-n can be, but not limited to a mobile device such as a cellular phone, a PDA and a smartphone, a laptop, a personal computer, and the like. Examples of application contents 106a-n include a game, a messaging, a calendar, an address, a notepad, a user interface and other applications to be executed on devices 102a-n.

[0024] Devices 102a-n may have different hardware and software platforms. Examples of software platforms include operating systems such as Windows, Linux, Macintosh, Symbian, and so forth. Moreover, devices 102a-n may have different hardware such as the screen size, screen resolution, audio and video functionality, processors and so forth. Therefore, development of an application for each of these platforms may require application content 106a-n to be written in various programming languages and functions compatible with the platforms. For example, C++ programming language may be used by Windows applications, while JAVA programming language is primarily used for developing web applets. Moreover, the resources required for each platform may be different. For example, the type and size of an image to be displayed may vary based on the screen size and screen resolution of devices 102a-n. Therefore, application contents 106a-n may be developed based on the platform of devices 102a-n.

[0025] Application development framework 104 provides a mechanism for developing application contents 106a-n for devices 102a-n independently across multiple device platforms. Therefore, application contents 106a-n developed using application development framework 104 is programmed only once and can be used on various platforms of devices 102a-n. In an embodiment of the invention, application development framework 104 enables application content 106a-n to be developed based on platform specific requirements and platform independent requirements. In another embodiment of the invention, application development framework 104 provides cross-translation of programming languages for various device platforms.

[0026] In case of conventional development or porting of applications, the number of coding modules for 'n' number of applications and 'm' number of devices can be n*m. For example, when 4 applications are developed for 10 different platforms of devices, then a total of 40 coding modules for applications have to be developed. As is evident from the above calculations, the number of coding modules increases tremendously with increase in the number of applications and the type of device platforms. However, in case of applications developed through platform-independent application development framework 104, the number of coding modules can be n*m. For example, the application logic can be written in platform independent code by using Application Programming Interface (API) provided by application development framework 104. Therefore, 'n' number of platform independent logics is required for 'n' number of applications. Similarly, using application development framework 104 can also program the platform specific programming code for 'm' number of devices. Therefore, 'm' number of platform specific coding modules is required for 'm' number of devices. As a result, a total of n*m. For example, when 4 applications are developed for 10 different platforms of devices by using application development framework 104, then a total of 14 coding modules for applications have to be developed. As a result, the complexity of the programming code and development time can be reduced.
Application development framework 104 uses abstraction layer and pre-processor functionalities to enable the programming code to be developed independent of the device platforms. The abstraction layer and pre-processor functionalities are described in detail in conjunction with FIGS. 2 and 3 respectively.

FIG. 2 shows functional overview of an abstraction layer 204 according to an embodiment of the invention. Generally, in the domain of software programming, an abstraction layer enables separation of various details or functionalities of a programming code from the device platform requirements. Abstraction layer 204 in the platform-independent development architecture separates platform independent code 202 from platform dependent codes 206a-b of an application. For example, platform independent code 202 may be the logic of the application and platform dependent codes 206a-b can be the programming code specific to devices 102.

Therefore, a developer that works on the logic or platform engine of the application may not require knowledge about the device interface. Moreover, the developer that works on the application itself may not require the knowledge about the device platform.

Platform independent code 202 and platform dependent codes 206a-b are developed by using API provided by application development framework 104. In an embodiment of the invention, platform independent code 202 and platform dependent codes 206a-b are parts of application content 106a-b. Platform dependent code 206a-b can be developed based on the native language of the specific platforms. The complete code of the application that contains the platform independent code 202 and platform dependent code 206a-b may be compiled to generate a binary codes 208a-b for the device platforms. Therefore, the application developed contains the complete code and any additional component or code is not required. Moreover, the processing overhead is low, because the application code is compiled into native binary code of the platform. In an embodiment of the invention, application development framework 104 implements abstraction layer 204 during the run-time of the application.

Abstraction layer 204 further provides infrastructure for the development of the application. The infrastructure provided by abstraction layer 204 includes various utilities, data structures, tools, design options, and resources required to develop the application. Examples of data structures and utilities include a set of collections like arrays and dictionaries, and examples of tools include Extensible Markup Language (XML) parsing, unicode string support and so forth. Examples of resources include images, sounds, videos and various other assets required by the application. In an embodiment of the invention, abstraction layer 204 includes different modules to develop the application. Examples of modules include, but are not limited to, networking, file support, rendering, resource loading, system events, system properties, and resource manipulation. Each of the modules provided by abstraction layer 204 include a common functionality, and a device platform specific functionality. The device platform specific functionality is implemented in the device plug-in and is explained in detail in conjunction with FIG. 7.

FIG. 3 shows functional overview of pre-processor functionality according to an embodiment of the invention. Generally, a pre-processor processes an input data to produce an output to be used as input to a computer program. In case of pre-processor manipulation, tags are embedded inside the application code. The tags instruct the compiler on the sections to be compiled for different platforms. Therefore, as shown in FIG. 3, application programming code comprises platform independent code 302 and that in turn comprises platform dependent codes 304a-b. The application programming code is compiled by a compiler to generate native binary codes 306a-b. Binary codes 306a-b can then be executed on the specific device platforms for which the code has been compiled. There are no layers between platform dependent codes 304a-b and platform independent code 302, therefore no runtime overhead is introduced in the execution of the programming code.

Application development framework 104 implements pre-processor for device platform specific compilation of application code. In an embodiment of the invention, pre-processor functionality is implemented to map logical resources and the real resources based on the device platform. As a result, the binary codes only include needed resources and therefore, the overhead for mapping is lower. Examples of resources include images, sounds, video and so forth. Moreover, the pre-processor functionality can be used to override implementation of application for a specific device inside the same device platform family. For example, a specific model of a mobile device of the Windows Mobile platform may have a different screen resolution for displaying images. Therefore, the Graphical User Interface (GUI) can be tailored for the specific device platform by using pre-processor functionality. Moreover, the pre-processor is used for translation of the programming code across different programming languages. For example, the programming code of a game may be for a mobile phone that only supports JAVA programming language.

FIG. 4 shows functional overview of platform-independent Application Development Framework (ADF) 104 according to an embodiment of the invention. ADF 104 comprises content interface 402, application environment 404, plug-in interface 412, plug-ins 414.

Content interface 402 provides the API for developing the application in ADF 104. Further, content interface 402 includes application specific logic, GUI logic, various assets such as images, sounds, fonts, texts, coordinates etc., and other optional modules. Content interface 402 enables the developer to select the modules for developing an application. The modules provided by content interface 402 are part of abstraction layer 204, and hence, they are independent from the platform and application. Examples of modules include but are not limited to, gaming module and social network module.

Application Environment (AE) 404 provides infrastructure to program the application. Examples of infrastructure include design options, utilities and tools that enable the creation of applications. AE 404 is a part of abstraction layer 204, and hence, is independent from the platform of devices. AE 404 comprises a utility module 406, a control infrastructure module 408, and an asset management module 410. Utility module 406 provides multiple utilities and data structures for developing applications. The utilities and data structures provided by utility module 406 can be used for platform independent application development of application logic, GUI and other parts of the application. Examples of utilities and data structures provided by utility module 406 include but are not limited to collections of arrays, vector, dictionary and the like, data types, GUI data types such as point, rectangle and the like, eXtensible Markup Language (XML) support,
Logic module 502 comprises the logic data of the application. The data of the application can be split among different objects, collections, and structures. Logic module 502 communicates with controller 506 and user interface module 504 to provide the data for the application. In an embodiment of the invention, logic module 502 corresponds to the model in the MVC architecture. User interface module 504 provides GUI to display a complete or partial data from logic module 502. Moreover, user interface module can display additional items, for example in the case of games, the animations and transitions.

Controller 506 controls the flow of the application. Controller 506 processes and responds to events, for example a user action. Moreover, controller 506 decides various events such as when to show a screen, when to change or update the logic module 502 and so forth. In an embodiment of the invention, control infrastructure module 408 may comprise multiple sub-controllers under controller 506. Each of the sub-controllers may be responsible for a single module or part of the logic. In an embodiment of the invention, sub-controllers can dispatch events to controller 506 that acts as a state machine. Subsequently, based on these events, controller 506 can start or stop sub-controllers or modules. User interface module 504 and controller 506 can communicate via asynchronous events. In another embodiment of the invention, events that are dispatched from user interface module 504 and from the network are handled by controller 506 and the sub-controllers. The sub-controllers can then decide to update the logic module 502, move to the next state in the application etc. Controller 506 and the sub-controllers can also dispatch events to the user interface module 504. For example, controller 506 may dispatch an event to start an animation. The use of events between the user interface module 504 and controller 506 provides better separation between their corresponding functionalities. Moreover, the developer of the application may use only logic model 502 and user interface module 504, Therefore, the developer may not program controller 506. As a result, the design of the application is cleaner and more flexible.

FIG. 6 shows functional overview of asset management module 410 according to an embodiment of the invention. The screens of different devices platforms can typically be categorized into computers, touch screens devices, or mobile devices with only key navigation. Further, each of the device platforms may have different screen resolutions. Asset management module 410 provides assets based on the device platforms and screen capabilities. Examples of real assets 608 include but are not limited to images, sounds, videos, and other GUI related requirements for a platform, and so forth. In an embodiment of the invention, different assets are split based on different screen families and resolutions to generate logical assets 604 for a GUI 602 by asset management module 410. The developer of the application only uses logical assets 604, Therefore, the real asset can be modified or tailored for a device platform, while the logical use of the asset remains the same. Further, asset management module 410 provides a mapping between logical assets 604 and real assets 608. Therefore, different platforms that share the same screen family or resolutions use exactly the same logical assets 604. As a result, the same application will look identical on both the platforms. However, the developer can customize the look for a specific device platform. In an embodiment of the invention, the developer can customize the GUI based on the device
platform fragmentations. Therefore, the flexibility in developing the application is increased.

[0042] Asset-mapping module 606 defines which logical assets 604 are used by each screen family or resolution combination. In an embodiment of the invention, asset-mapping module 606 includes a set of scripts for mapping logical assets 604 to real assets 608. The scripts provided by asset-mapping module 606 are managed through a GUI editor. In an embodiment of the invention, the scripts provided by asset-mapping module 606 are managed manually. In an embodiment of the invention, logical identifiers are associated with logical assets 604. Asset-mapping module 606 maps logical assets 604 to real assets 608 based on the logical identifiers. Examples of logical identifier include the name, location, size, or characteristics of logical assets 604.

[0043] In an embodiment of the invention, GUI customization of real assets 608 is done in the pre-processing stage. As a result, runtime overhead for executing the application is less. However, parts of the GUI can also be customized during runtime. For example, the application can ask AE 404 for the platform resolution, family, orientation etc., and then make decisions based on it. In an embodiment of the invention, real assets 608 are maintained in a single lossless format. For example, images are maintained in a PNG format. Real assets 608 are converted to platform specific format during the pre-processor stage. Therefore, no runtime overhead is encountered. Moreover, multiple real assets 608 in different formats for the same screen resolution are not required to be maintained.

[0044] FIG. 7 shows exemplary functional overview of plug-in interface 412 according to an embodiment of the invention. Plug-in interface 412 is used when platform specific functionality is required for an application.

[0045] Exemplary flow for reading a number of bytes of data from a web server is illustrated in FIG. 7, to explain the function of plug-in interface 412. In an embodiment of the invention, the files module is used to provide network connectivity. The file module has two main components: an abstract functionality 702 and a plug-in platform specific functionality 704. Abstract functionality 702 includes various functions that are independent of the device platform. For illustration purposes, in case of networking the functions in abstract functionality 702 include connect-server function 706, read bytes function 708, and disconnect-server function 710.

[0046] As shown, connect-server function 706 interfaces with platform specific functionality 704 to create a system socket and attempts to make a connection. Similarly, read bytes function 708 interfaces with platform specific functionality 704 to read bytes from the socket. The disconnect-server function 710 disconnects and releases the system socket. The developer may use only abstract functionalities 702 of ADF 104, which are automatically translated to platform specific functionalities 702.

[0047] FIG. 8 is a flowchart illustrating the method for platform independent development of an application, according to an embodiment of the invention. At step 802, an Application Programming Interface (API) is provided to program an application comprising a programming code. The programming code can be executed on multiple platforms. In an embodiment of the invention, the multiple platforms include mobile device platforms. Examples of the application include, but are not limited to, an online game for mobile devices, a messaging application, a chat application and so forth. The API is independent of device platform and is provided by ADF 104. Thereafter, at step 804, application environment 404 is provided to program the application. Application environment 404 provides functions and infrastructure that is independent of the device platforms. Infrastructure includes various utilities, data structures, tools, modules, assets and other resources or modules required for developing or executing the application. Subsequently, at step 806, an interface is provided between application environment 404 and the device platforms. In an embodiment of the invention, the interface is provided by plug-in interface 412.

[0048] Embodiments of the invention are described above with reference to block diagrams and schematic illustrations of methods and systems according to embodiments of the invention. It will be understood that each block of the diagrams and combinations of blocks in the diagrams can be implemented by computer program instructions. These computer program instructions may be loaded onto one or more general purpose computers, special purpose computers, or other programmable data processing apparatus to produce machines, such that the instructions which execute on the computers or other programmable data processing apparatus create means for implementing the functions specified in the block or blocks. Such computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means that implement the function specified in the block or blocks. Furthermore, such computer program instructions may be made available for download and/or downloaded over a communication network.

[0049] While the invention has been described in connection with what is presently considered to be the most practical and various embodiments, it is to be understood that the invention is not to be limited to the disclosed embodiments, but on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

[0050] This written description uses examples to disclose the invention, including the best mode, and also to enable any person skilled in the art to practice the invention, including making and using any devices or systems and performing any incorporated methods. The patentable scope the invention is defined in the claims, and may include other examples that occur to those skilled in the art. Such other examples are intended to be within the scope of the claims if they have structural elements that do not differ from the literal language of the claims, or if they include equivalent structural elements with insubstantial differences from the literal languages of the claims.

We claim:

1. A platform-independent application development framework, comprising:

a content interface configured to provide an Application Programming Interface (API) to program an application, the application comprising programming code to be executed on one or more platforms, wherein the API is independent of the one or more platforms;

an application environment configured to provide an infrastructure to program the application, wherein the infrastructure is independent of the one or more platforms; and
one or more plug-in interfaces configured to provide an
interface between the application environment and the
one or more platforms.
2. The framework of claim 1, wherein the content interface
is further configured to provide one or more assets for pro-
gramming the application.
3. The framework of claim 1, wherein the application envi-
ronment comprises:
a utility module configured to provide one or more
resources to program the application;
a control infrastructure module configured to manage the
flow of the application; and
an asset management module configured to manage one or
more logical assets.
4. The framework of claim 3, wherein the one or more
resources comprise one or more data structures.
5. The framework of claim 3, wherein the control infra-
structure comprises:
a model module configured to provide the data of the
application;
a view module configured to provide a user interface for the
data provided by the model module; and
a controller configured to handle one or more events
received from the model module and the view module.
6. The framework of claim 5, wherein the control infra-
structure further comprising one or more sub-controllers.
7. The framework of claim 6, wherein the one or more
sub-controllers are controlled by the controller.
8. The framework of claim 6, wherein the one or more
sub-controllers are configured to handle the one or more
events received from the model module and the view module.
9. The framework of claim 6, wherein the one or more
sub-controllers are configured to dispatch the one or more
events to the controller.
10. The framework of claim 3, wherein the one or more
assets provided by the asset management module are inde-
pendent of the one or more platforms.
11. The framework of claim 3, wherein the asset manage-
ment module is further configured to provide one or more
logical asset identifiers to the one or more logical assets.
12. The framework of claim 3, wherein the asset manage-
ment module is further configured to map the one or more
logical assets to the one or more real assets based on the one or
more logical identifiers.
13. The framework of claim 1, wherein the one or more
plug-in interfaces are configured to compile the program code
to one or more native binary formats corresponding to the one
or more platforms.
14. The framework of claim 1, wherein the application
environment is further configured to emulate the application
on the one or more platforms.
15. The framework of claim 1, wherein the application
comprises the programming code of a mobile game.
16. The framework of claim 1, wherein at least one of the
one or more platforms is a mobile device platform.
17. The framework of claim 16, wherein the application
comprises the programming code of a mobile game.
18. The framework of claim 16, wherein the application
environment comprises:
a utility module configured to provide one or more
resources to program the application;
a control infrastructure module configured to manage the
flow of the application; and
an asset management module configured to manage one or
more logical assets for the at least one mobile device
platform.
19. An architecture for platform-independent development of
an application, the application comprising a programming
code to be executed on one or more mobile device platforms,
the architecture comprising:
an abstraction layer configured to:
provide an Application Programming Interface (API) to
program the application, wherein the API is indepen-
dent of the one or more mobile device platforms;
provide an infrastructure to program the application,
wherein the infrastructure is independent of the one or
more mobile device platforms; and
configured to provide an interface between the infra-
structure and the one or more mobile device plat-
forms.
20. The architecture of claim 19 further comprising an
emulation layer configured to emulate the application on the
one or more mobile device platforms.
21. The architecture of claim 19, wherein the plug-ins layer
is further configured to compile the programming code of the
application based on the characteristics of the one or more
device platforms.
22. The framework of claim 19, wherein the abstraction
layer comprises:
a utility module configured to provide one or more
resources to program the application;
a control infrastructure module configured to manage the
flow of the application; and
an asset management module configured to manage one or
more logical assets for the one or more mobile device
platforms.
23. A method for platform independent development of an
application, comprising:
providing an Application Programming Interface (API) to
program an application, the application comprising a
programming code to be executed on one or more plat-
forms, wherein the API is independent of the one or
more platforms;
providing an application environment to program the
application, wherein the infrastructure is independent of
the one or more platforms; and
providing an interface between the application environ-
ment and the one or more platforms.
24. The method of claim 23, wherein providing the API
comprises providing one or more assets to program the applica-
tion.
25. The method of claim 23 further comprising emulating
the application on the one or more platforms.
26. The method of claim 23, wherein generating the pro-
gramming code comprises providing one or more modules for
programming the application.
27. The method of claim 23, wherein generating the pro-
gramming code comprises providing one or more logical
assets.
28. The method of claim 23, further comprising downloading
computer-executable instructions that, if executed on a
computer, cause the computer to implement said providing an
API, said providing an application environment, and said
providing an interface.
29. The method of claim 23, further comprising providing
downloading computer-executable instructions that, if executed on a computer, cause the computer to implement
said providing an API, said providing an application environment, and said providing an interface.

30. A computer-readable medium having computer-executable instructions that, if executed by a computing device, cause the computing device to implement a method for platform independent development of an application, comprising:

- providing an Application Programming Interface (API) to program an application, the application comprising programming code to be executed on one or more platforms, wherein the API is independent of the one or more platforms;
- providing an infrastructure to program the application, wherein the infrastructure is independent of the one or more platforms; and
- providing an interface between the application environment and the one or more platforms.

31. The computer-readable medium of claim 30, wherein providing the API comprises providing one or more assets to program the application.

32. The computer-readable medium of claim 30, wherein generating the programming code comprises providing one or more modules for programming the application.

33. The computer-readable medium of claim 30, wherein generating the programming code comprises providing one or more logical assets.

* * * * *