A Content Adaptation Architecture for Games

Mark Joselli, UFF, IC, Medialab
Jose Ricardo Silva Junior, UFF, IC, Medialab
Marcelo Zamith, UFF, IC, Medialab
Esteban Clua, UFF, IC, Medialab
Eduardo Soluri, Nullpointer Tecnologias, http://www.nullpointer.com.br

Abstract

Most of the game content is designed specificaly for the device, which it is being developed for. Nowadays, with the development of mobile phones, smartphones, tablets and Digital TV, this content must be adapted to these different devices, where each have diverse characteristics and constraints, like different screen dimensions and multimedia capabilities. The adaptation task can consume time and resources, since most solutions resolve it by creating different resources for different devices. Also these resources and different versions of the resources must be managed in a higher level. In this work, a novel content adaptation architecture is presented, which was developed to gather the content from a content management system (CMS) or a version control systems (VCS) and adapt it for different devices, through the use of templates, which describes how the content must be adapted. Furthermore, as most games have different versions and content versions, like levels and characters, the architecture have a content version control, where the client (the device the game is installed and played) only receives the difference between the content version that is installed and the downloaded one. This way, the architecture avoids higher data transfers thought the network, which in some cases can be slow and expensive, since some devices connect though slow and expensive mobile networks.

Keywords:: Content Adaptation, CMS, Game Backend

Author's Contact:

mjoselli@ic.uff.br
jricardo@ic.uff.br
mzamith@ic.uff.br
esteban@ic.uff.br
esoluri@nullpointer.com.br

1 Introduction

Ubiquitous computing [Holloway and Julien 2010] presents a new computational paradigm, where information processing is integrated into everyday objects and activities. Among relevant characteristics of this kind of computing, one is that the different devices that use this technology have different characteristics, requiring different types of media. These devices are Smartphones, tablets, e-readers and Digital Tvs [Joselli et al. 2012b].

Mobile devices, like smartphone and tablets, and Digital TVs have many different characteristics [Joselli and Clua 2009; Joselli et al. 2012a], when compared to the PC, like: hardware constraints (processing power and screen size); user input, (buttons, voice, touch screen and accelerometers); and different operating systems, like Android, Blackberry OS, iPhone OS, Symbian and Windows Mobile [Joselli et al. 2012c]. All these different characteristics must be considered when the game content needs to be delivery for these devices [Hildebrand et al. 2007], thus the design and adaptation of content for multiple platforms and devices becomes extremely difficult The presented architecture provides a Content Adapter Architecture (CAA), which can be used to adapt all content for mobile devices.

In general, the content adaptation envelops not only the adaptation of format and types, but also different styles, dimensions, data compression and specifications [Chaari et al. 2007] [Carvalho and Trinta 2009], since quality of the user experience can suffer from a non adapted (or poor adapted) media [Agboma and Liotta 2010]. Also, different contents and information can be available for specifics devices or systems, requiring a custom adaptation or generation of the content, which the CAA can provide.

Games are multimedia applications [Ricardo da Silva Junior et al. 2012], handling a lot of game content that needs to be manageable. The game content can be special downloadable content, like videos, wallpapers and documents; extra content as new characters, levels, even upgrades of the game; and also content for its web site. Having this many content types requires a content manager. Games now deal with different platforms and versions like web games, social games, console games and mobile games [Joselli et al. 2010], and most of the time it needs special contents for these different versions. Normally, all kinds of games deal with content and need some form of management. The proposed CAA also provides a form of managing all game content.

2D games need different kind of images for different screen resolutions. Nowadays, Android and IOS devices can have multiple content for different resolutions in the same application/game. In the Android, this content is divided by screen density. IOS divides by iPhone or iPad and normal or retina displays. Though this solves the problem, the game or application size increases considerably by using such approach, since it keeps both copies inside the application package. The architecture presented in this paper could be used in order to have only the needed and right resource for each device. 3D games could also use this architecture, since more powerful devices can have models with more polygons in the scene, CAA can manage and deliver different models for different devices.

Most of the World-Wide Web's content is managed through CMSs (content management systems). Some games use CMS's features to handle part of its contents. The CMSs provides a better content organization, increased access to resources and greater organizational effectiveness, among many of its advantages. CMSs can be used, natively or with plug-ins, for the management of game content, to enable different devices, like TVs and mobile devices, which can require new contents. A Version Control System (VCS) administers changes of documents, programs, images and other information stored in form of computer files. It can be used to control code revision, and also to manage resources, updates and releases of the game. The Content Adapter Architecture provides an adaptation of the resources from CMSs or VCSs for different devices with the use of templates, providing a generic non-intrusive content adaptation. Also, natively, CMSs and VCSs do not provide cache mechanism, compression of content, and version control (for the content on the client), which the CAA provides.

In mobile devices the connectivity can still be very slow and expensive, depending on the network connection and the network carrier, requiring some form of data compression and caches techniques to reduce the data transfers. The CAA provides a cache mechanism in order to fulfill these requirements. A version control of the content is also provided, so that the content transferred between the server and the client, the game, is only the difference between the data that is on the device and the new data from the server.

The Content Adapter Architecture consists in two modules: a server and a client plug-in. The server is responsible for: adapting the content from different sources through the use of templates; compressing and applying cryptographic algorithms to protect adapted content; delivering content to the device; and controlling the different contents versions. The client plug-in function as a layer on the game, which is able to open the package, sent by the CAA, and to organize the content into the game.

Summarizing, this work provides an architecture that has the following features has been developed:

- ability to gather game content from different sources, like CMSs and VCSs;
- adaptation of this content for different devices characteristics;

- creation and control of content versions;
- collection of statistic data, in order to create users profiles;
- cryptography system to protect the game content;
- data compression for minimizing network data exchange;
- and a thin client plug-in for the content organization and server communication has been developed.

This work is divided as follows, Section 2 presents the related works. Section 3 presents the Content Adapter Architecture server and Section 4 its client. Section 5 shows the test case in order to validate the presented architecture and finally, Section 6 presents the conclusions and future works.

2 Related Work

CMS can be used as content provider for different devices. This can be done in different ways, but it normally requires publishing of different versions of content for different platforms. Another common solution is the development of plug-ins or add-ons for the CMS that could achieve the same effect by adapting the resources. Nurminen et al. [Nurminen et al. 2008] show some experience on the use of Drupal CMS with some add-ons for mobile website provisioning. The drawback of such approach is that this plug-ins and add-ons function for a specific CMS, the CAA can do the same adaptation in a generic way, working with any kind of CMS.

Some companies have developed CMS specific for mobile platforms, like Magnolia CMS [Internation 2012], Cellular CMS [Services 2012], Mobile CMS [MobiManage 2012] and mFabrik [mFabrik 2012] to name a few. They work like traditional CMSs that are customized for the mobile content. Manashty et al. [Manashty et al. 2010] presents the ARMrayan MCMS a CMS designed for J2ME mobile applications. Moreover, the work Hildebrand et al. [Hildebrand et al. 2007] presents a CMS platform for e-learning specific for mobile devices. The downside of such approaches is that these CMS are normally specific for providing content for mobile sites and they are not prepared to work with different game content like levels and 3D models.

There is also some work into providing a midlleware to adapt the content. Trinta et al [Trinta et al. 2007; Trinta et al. 2008] presents a middleware to support multiplayer multi-platform games. Among services it provides, there is a content adaptor, which transforms the information of the game according to the actual context of the player. Its adaptation is of the game information, and not the game content, as this work approaches. Another middleware service is the work Ubidoctor [Diniz et al. 2008]. The Ubidoctor is a middleware for the construction of ubiquitous application for the medicine area. It provides the service of session management, context management and content adaptation. The content adaptation transforms the resources for the ubiquitous healthcare applications, but seems very simple adaptation and appears to be limited by the type and sizes, a limitation CAA does not have.

Delicato et al. [Delicato et al. 2009] and Tummala and Jones [Tummala and Jones 2005] work on context and location awarenes, where specific CMS or middleware are developed for this purpose. In this kind of application the content delivery for the end-user is dependent on his context or location. The CMS Adapter Architecture could also be used for this purpose, since its templates can require specific data from the CMS and the thin client can provide information about localization and send it to the CAA that would then delivery the content for the matched location or context, but the architecture still needs some testing and adaptation in order to fully provide this service.

One main disadvantage of so far presented works is the lack of generalization allow use in others contexts, like different content providers. One different approach comes from the CAS service [Carvalho and Trinta 2009][Diego Carvalho 2009] that provides a content adaptation as a generic service performing adaptation of text, audio, video and image for multi-platform applications. CAS built its services modular and non-intrusive, similar to the CAA, but its lack of some of here proposed features, like the cache, data compression and version control of content.

Further there is research on the adaptation of game content, [Morán et al. 2007] shows the adaptation of models in a online server for different devices. Also the work [Yannakakis and Togelius 2011], shows the generation of content for different platforms using procedural content. These work are being considered for the 3d content adaptation, but they still lacks of generic adaptation of the model content normally used by the mobile devices, as this work is trying to achieve.

One of CAA's features is the use of cache. On this field, the work by Li et al [Li et al. 2009] presents a statistical mechanism for maintaining cache consistency in a mobile environment. It tries to deliver a new cache mechanism since most existing cache consistency strategies assume reliable communication between mobile terminals, and cannot handle frequently offline devices adequately (a common situation for mobile devices). This work also uses cache for the application content, and also tries to avoid inconsistency in the cache, but the architecture uses a simpler mechanism for doing that, requiring the verification of the resources hash code.

3 The Content Adapter Architecture Server

This section is devoted to explain how the backend of the Content Adapter Architecture works. It is divided in four subsections, which are core elements of the architecture are shown: the controller, the content adaptation, the content version control and the identity server.

In order to work with a game, this architecture uses a CMS to help managing and building the deliverable game content to the game. Content also can come from version control systems (VCS), like Git, CVS and SVN. Each content, depending on the template, is delivered or adapted to the device.

Also, as game users are spread across the world, the architecture can be used to have the deliverable content on the cloud, in order to increase service efficiency to the enduser.

CAA is build as a web service that adapts and creates content from CMS and version control systems based on templates. An overview of the CAA server can be seen on figure 1.

This figure presents the architectures main modules. The external modules are also shown: CMS and / or VCS, they are the resources providers. All content is gathered by the controller, through the use of connectors, and saved to a database. The controller is the service connecting all the other ones, making them work together. Next, the resources are adapted through the use of templates and saved on a proprietary or cloud server, like the Amazons S3 and Cloudfront. Further, resources from secure services that need authenticating and secure connection, like VCS or protected CMS, can be delivered and adapted, using the identity server. Push notification services, which are normally on the brand server, are accessed via a web service, and triggered by the controller when new content arrives.

CAAs templates for customization and configuration have to be registered in the controller by a developer/publisher/editor. These templates are built upon an XML based language, and they are the bases for how the Content Adapter Architecture adapts the contents. Also, some content can be device specific, like icons, logos and even 3D models. In this case, the content can be on the CMS server, on the VCS server or it can be in the Content Adapter Architecture.

The architecture is built upon components. Based on previous experience in using this model in different application domains. A variety of technical implementations were used to abstract and create a general architecture where different methods for content adaption, personalization and contextualization can be achieved.

The architecture server is responsible for:

- gather of content from the registered CMSs or VCSs;

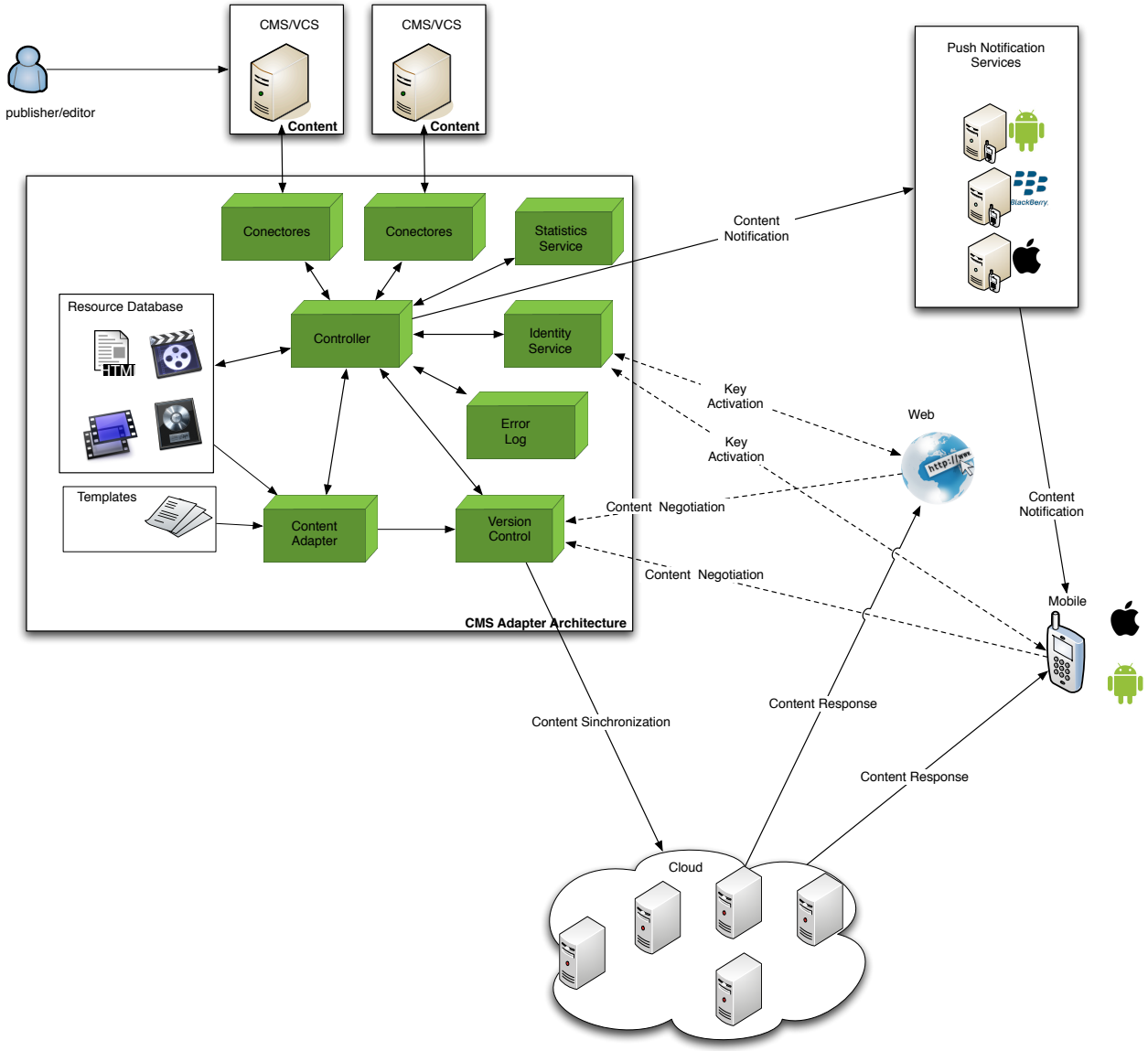


Figure 1: *Overview of the Content Adapter Architecture server.*

- adapt content for the different platforms according to templates;
- implement a AAA server (Authentication, Authorization e Accounting) in order to implement a security server to communicates with servers that needs secure policy, such as, login to secured CMSs server or SVC for the content gathering;
- manage of the user/game statistical data, collecting it from the devices and saving it for analytical packages;
- send push notification to the end-user as necessary (when configured to do so, like when new content becomes available);
- cryptograph content based on public and private keys, if needed;
- keep a version control of the content on each user, and send the new content to the user;
- compress the content to send to the user;
- provide statistic reporting from the service side, plus information gathered by the client.

The architecture has an error log in the server, which is responsible for gathering any error or warnings that can come from the resource collection, adaptation of content or the generation of a new version. Errors can come from different facts, like trying to gather a resource that is not on the server or adapting a content that cannot be adapted to the description that is on the template. Error logs can be delivered to registered emails, or be seen on web reports.

The server also receives statics data from the game, and saves it on the statistics services module for later use on web reports. These statistics can be used to analyze the architectures performance and the players use of the game.

3.1 Controller

The controller is the main component of the architecture. Its main responsibility is to orchestrate all the tasks and processes from all others components.

The controller request content from the CMS, which are connected though connectors, and then saved to the resource database. The resources are sent to the content adapter, which convert the data with the use of templates. The adapted version is saved on the version control module, which generates a new version, by compressing it and generating a new version ID. Then this version is saved on the cloud. This process is better illustrated on the Figure 2.

The publication of the content for the device can be done in two

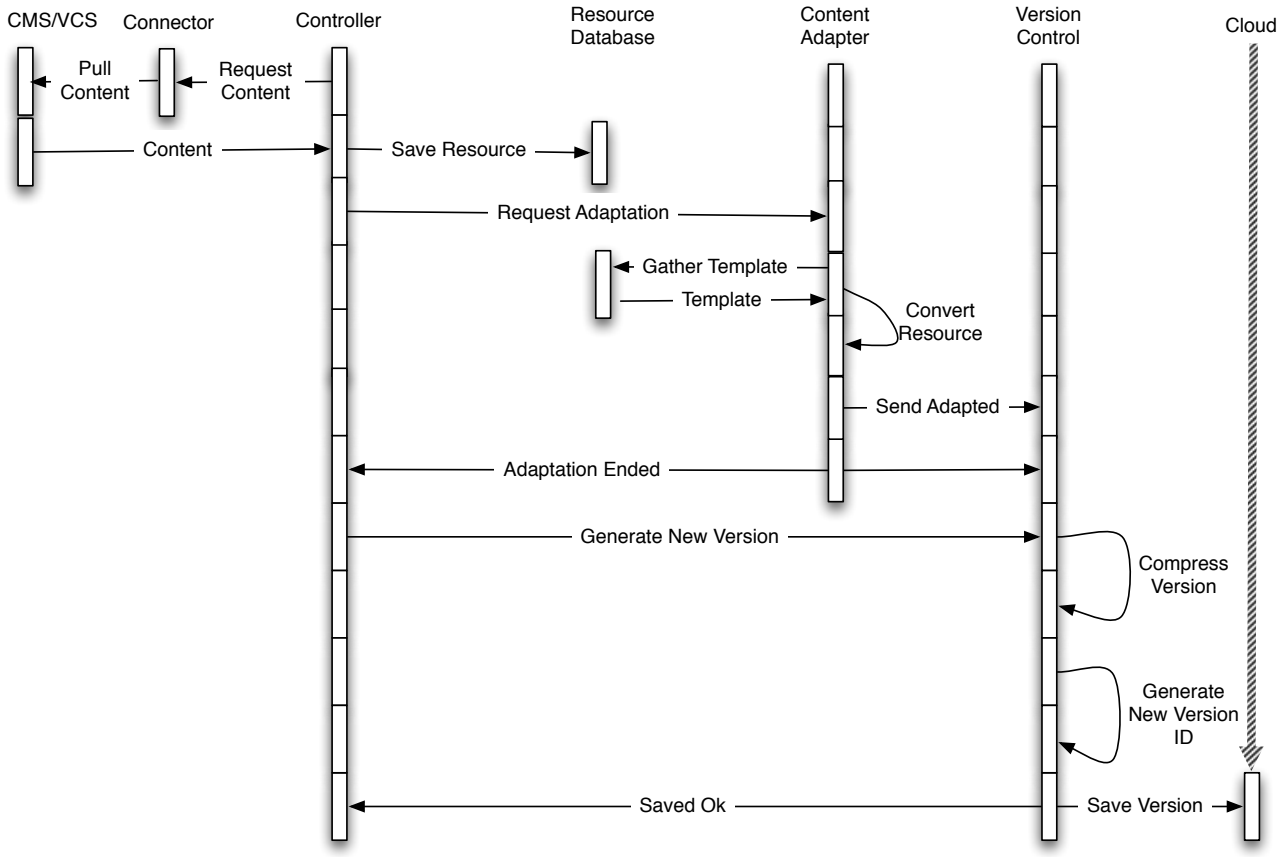


Figure 2: *Overview of the Execution of the Architecture.*

ways: via content aggregation (pull); and content syndication (Push). Content aggregation is the ability to pull content from CMS or VCS and adapt it though templates. Content Syndication is done mainly by the user in their CMS/VCS and requires the republishing of the content for the required device, in this way this content is only copied to the CAA repository. The content syndication is mostly used on interface content, like icons, models and logos.

A scheduler pattern is used in order to call each CMS/VCS service, when they are configured as a timer for content gathering. Furthermore, a command pattern is used in order to encapsulate all the information needed for the service to be called, when the time comes.

```
<Config>
  <Contents cms='Joomla'>
    <content type='automatic'>

      <service timer='2h'>
http://201.200.1.132/levels.php
      </service>

      <parameters>

        <param type='datetime'>
date
        </param>

      </parameters>

      <template>
          levels.template.xml
      </template>

    </content>
  </Contents>
</Config>
```

With above the xml, the CAA gathers information from a Joomla CMS using a scheduler of two hours, i.e., it will run every two hours to check for new content. When new content is available, It will gather the content using the http://201.200.1.132/levels.php web service, which requires the datetime as an input, a system variable that holds the time when the last request was made. Also, as it can be seen in the XML, the Contents tag carries all the contents with their description, and shows how each should be adapted for the desired platform, using another XML. For the content adaptation, it will use a template which is defined in the file levels.template.xml, which is described in the next subsection.

3.2 Content Adapter

The content adapter is responsible for gathering the content from the resource database and adapting it to the required devices. This component normally uses a XML for the configuration of the adapter, and a series of XMLs describing how each of the content should be adapted. The process is simple, it uses a XML to gather information about the service that provides the content, and how it should gather and adapt these content.

The component can gather the information in two ways, via a push notification made by the CMS/VCS service, or in automatic manner using a timer. The push notification requires the service to be implemented in the CMS/VCS, but it has the advantage that the content can be adapted as soon as published. In the automatic manner, this implementation is not required, but the content is only published for the media when the service runs.

This service can support different contents, like sound, music, images, documents, video and HTMLs. The adaptations of these content are executed using templates, which are created as a xml descriptor. These templates are implemented using the template method design pattern, which defines the program skeleton of an algorithm, in this case the methods needed for the content adaptation.

An example of this XML for image adaptation can be seen bellow. In this XML, it uses an image adaptation in order to present a 320x240 image for the end-user.

```
<Template>
        <method>imageResize</method>
        <input type='image'>imageSrc</input>
        <algorithm>bilinear</algorithm>
        <output>
                <type>
                image
                </type>

                <format>
                png
                </format>

                <dimension>
                320x240
                </dimension>

                <quality>
                high
                </quality>

                <alpha>
                true
                </alpha>

        </output>
</Template>
```

The XML above will use an imageResize method to adapt the resource using the bilinear scaling algorithm and it takes as input an image. As the output it will provide an image with the following characteristics: a png format with 320x240 of size, with high quality and alpha channel.

There is a series of templates already built as components for the architecture, like images, audio and video. Sometimes, different devices implements different multimedia capabilities and also different game engines and frameworks uses them differently. Therefore, some devices may need for different media format to be able to use it. The CAA provides the adaptation of format types and also the images proprieties. The imageResize could also be used for cropping and composing images. This is very important especially in games for the creation of animation sprites.

The image adaptation is normally done by the ImageMagick library [1], a free open source library, used by CAA uses to change the image properties, like dimension, format and qualities. For the adaptation of audio or video, the architecture uses the FFMEPG library [2], another open source library,which converts the video/audio files. Thus, the adapter can change content properties, like the frame rate, format type, quality and dimensions. The configuration for each common device (mobile device) is registered in the application.

There are also adaptations of HTML pages for the devices. This is achieved with templates, where some tags are selected and others adapted, also the CSS and Javascripts files are sometimes modified or replaced in order to achieve a better final result.

These classes was implemented as a factory pattern, like figure 3 illustrates, where it takes a xml as input, and gives the object with the characteristics of the generated new resource as output. This way the architecture keeps the code clean by creating objects without specifying the exact resource class of object that will be created.

In this UML, there are the main classes of content adaptation. The image class is responsible for converting the images and it has the following childs: the sprite generator, which creates a sprite though combining a series of images into one single image that can be used for 2D animations; the Image2square, which is created by inserting blank pixels on the image until it has the dimension of power of 2, required for the use on some mobile openGL devices; the image aspectRatio, which converts the images, with different aspect radio, by resizing it and inserting or removing transparent pixels of the image; the Crop Image, which crops the image to a different size; and the imageResize, which resizes the image according to a scale factor. There is also a video adapter, which converts the video dimensions, bit-rates, codecs and encapsulation. And the text adapter for the conversion of text, by removing and changing characters, from different char-sets. The HTML adapter transform html to different platforms, by removing features or even changing the code. The sound adapter converts the sound from different bit-rates and different formats.

3.3 Content Version Control

The content version control is responsible for keeping the control of each version of the adapted content. Every time the CAA gathers content from the CMS/VCS or when its templates are changed, version control generates a new version on its repository. This component was built using a balk pattern, i.e., a design pattern that only executes an action on an object when the object is in a particular state, in this case it only updates and generates a new version of the content, when all the content gathering and adaptation has finished doing its work.

Since the content can chance frequently, and each user can have different versions, when the user updates its contents, he uses the version number that was saved on the cache, so that the client only downloads the needed content once for the up-to-date version. This way, the architecture tries to keeps the communication between the server and device to a minimum.

All the content generated by the content adaptation, is normally saved on the cloud, using Amazon Simple Storage Service, or similar services for cloud storage.

Likewise, all content sent to the user is compressed in a zip format by using the zlib library [3]. Also contributing for the communication between server and client to be kept at minimum, which is good for mobile devices that sometimes suffer from slow network and expensive data plans. This package can also be secured by running a cryptograph algorithm using the private key on the application.

3.4 Identity Server

This server of the architecture is used for different things: an AAA server (Authentication, Authorization and Accounting); a certification generator and manager for security connection between devices and the service; protecting special content, by doing cryptography on the files; and a statistics repository.

Based on the concept of Authentication, Authorization and Accounting, this component is responsible for implementing the security requirements for data exchange and consumption of restricted provided content between clients, the devices, and the servers.

Being an Identity Server, it is responsible for provisioning the user and its devices access to available content. This also requires the creation and management of certification keys and tokens when content cryptography is required.

Additionally, this server gathers statistics information for each device, and each user, storing content version, device capabilities and how the user uses the content. This server can provide real-time statistic reporting, grouping data by user, device or time.

4 The Client Plugin

The client plugin is responsible for: gathering the data from the CAA server; providing the server with devices characteristics; implementing a cache system for games gathered data; and collecting of user data for statistics reports.

The client plugin is developed as a framework, in order to be reused in others games. To work with the apple IOS and Google Android

[1] http://www.imagemagick.org/

[2] http://ffmpeg.org/

[3] http://zlib.net/

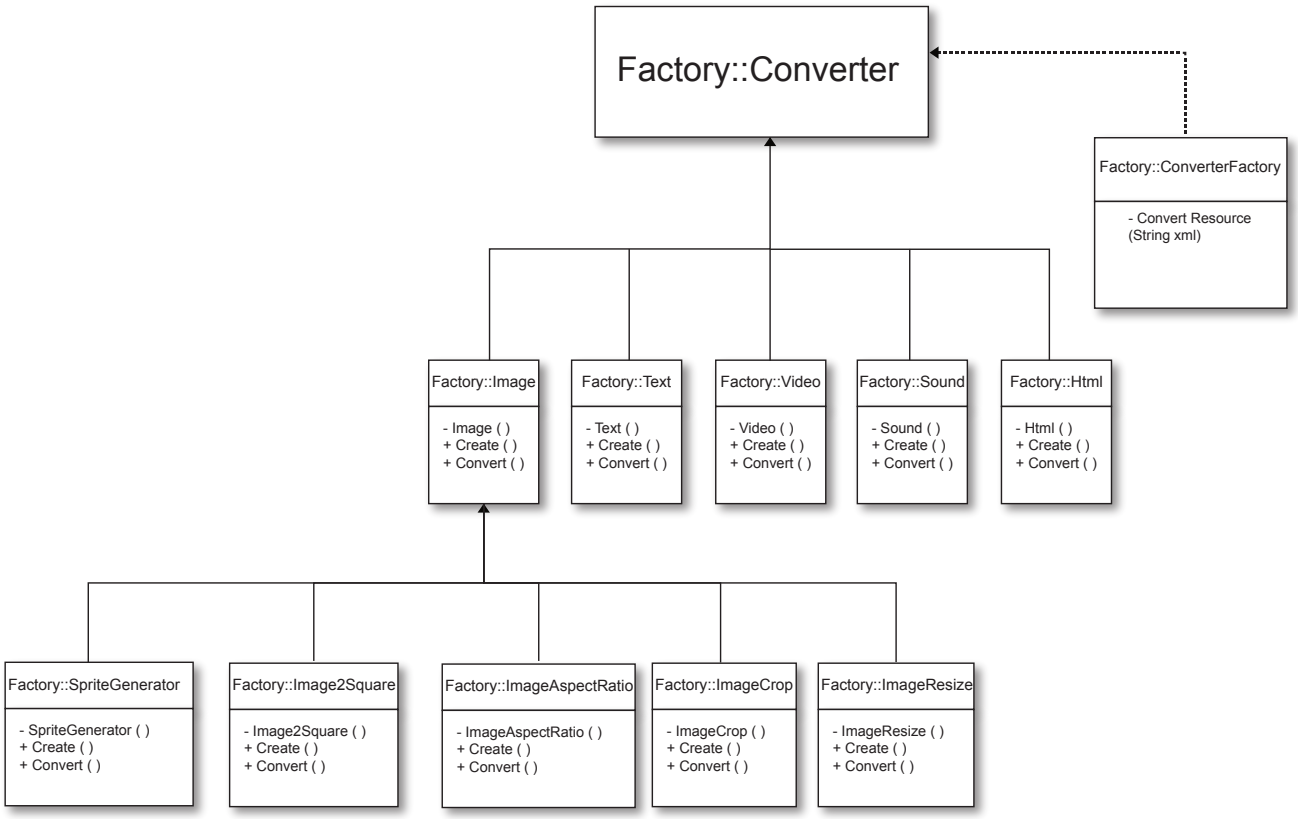


Figure 3: *UML of the Content Adapter.*

platforms, it is basically developed in object-C and Java. Frameworks are being developed for others platforms, such as Windows mobile, Blackberry and J2ME.

Mostly of the customization in the application is done via XMLs, such as the XML bellow, which configures how the application will download the data and where it will save it.

```
<ApplcationPush>
    <server timer='2h'>
     http://201.200.1.210:8000/ContentArch.js
    </server>
    <parameters>

            <param type='datetime'>
             date
            </param>

            <param type='model'>
             iPhone
            </param>

            <param type='gameengine'>
             cocos2d
            </param>

            <param type='cryptografy'>
             true
            </param>
          </parameters>

       <cache>
             <maxSize>
              100MB
             </maxSize>

             <diretory>
             APPDIR/ContentArch/Cache/
             </diretory>

             <param type='cryptografy'>
              false
             </param>

       </cache>

       <statistics>
            <timeBetweenUpdates>
             true
            </timeBetweenUpdates>

            <resourcesViewed>
             true
            </resourcesViewed>

            <dateTimeAppUse>
             true
            </dateTimeAppUse>

       </statistics>

</ApplcationPush>
```

The XML above will configure clients service for data, cache and statistics gathering. The XML sets the application to call the server every two hours (only if the application is running and has a network connection) on http://201.200.1.210:8000/ContentArch.js web service that requires a datetime as input, the device model (iPhone), the game engine used (cocos2d) and a variable saying that it will not require cryptography. This XML will also configure the cache use that will have a 100 MB of maximum size, no cryptography and will use the directory APPDIR/ContentArch/Cache/ to save the contents. The statistic gathering is configured by having the XML monitoring the time between updates required by users, the resources users view and the date/time of users interactions with the application.

The AAA (Authentication, Authorization and Accounting) client is responsible for the implementation of user identification procedures and the establishment of his permissions. It supports security communication (through https protocol), cryptography / encryption and

decryption.

Additionally, the client plug-in provides a services locator, where all the device capabilities are mapped in order to provide the best available content for the end-user. These capabilities can be device model, screen size, location services, game engine used, and types of available inputs, like a camera, plus any other device characteristic available for the client.

The client uses cache extensively for its contents. It uses this cache while providing for the end-user in order to minimize the data transfer between updates. This cache can also be cryptographically encoded or compressed if needed by the application. Further, as cache consistency can be a problem in some devices, when network data exchange fails [Li et al. 2009], This client only updates its version after gathering all the data, and verifying its hash code between the file downloaded and the sender by the server. It also uses a service to save the sessions data, gathering applications use statistics, if needed by the server.

To update data in the client, the following workflow is used: first, the device, if registered, gets a push notification of new content availability; next, when the user opens the application, the device authenticates and gathers needed resources for its update from the server; it the user has no available connection, the application starts without this update process, by accessing the cached resources. Figure 4 illustrates this process.

5 Validation

In order to validate the architecture, the CAA was used in a Pac-man clone that were developed for the Web, iPhone, iPad, Android's phones and tablets.

In order to run on all these different platforms, all the images must be adapted for the different screen sizes and technologies. Figure 5 shows some of the adaptation made by the CAA.

As illustrated on the image 5, the CAA takes a sprite sheet as the original image, and crops the first line of the sprite on eight different images. It also takes these eight images and creates a new sprite sheet. The first cropped image is used to be resized to half of its' original size, and also to three times its' original size. The scaled version (3x) is also used to generate a new image in a two square dimension.

In this validation scenario all the images were gathered from a CMS, a Joomla!. It is used as push service, where the content is gathered and converted as sons as a modification is signalized. The Content Adapter Architecture architecture saves some rework when customizing difference contents releases for different platforms.

The server was implemented on a Linux with Ubuntu 12.04 LTS, running an Apache with a MySQL database. The web services were developed using PHP. The Amazon S3 was used as the cloud storage. The game was developed using HTML5 with the logic on the javascript using the akihabara [4], which is a set of libraries, tools and presets to create web games.

This application also gathers some statistical data which is sent from time to time to the server to usage analysis. This information can be used later for promotion, sales and other events within the game.

Samples of the Game running on different platforms can be seen on figure 6. In the figure (a) can be seen the game on the web site, in (b) in a smart phone and in (c) on a tablet. From these images, can be seen that different layouts and images sizes can be adapted by using the presented architecture.

6 Conclusions

The devices that run games have many different characteristics and constraints, requiring specific content nowadays. Therefore, it is necessary to publish specify content or provide for adaptation of already published content according to devices needs. This paper has presented a new architecture for content adaptation, creating a layer between the content and the devices.

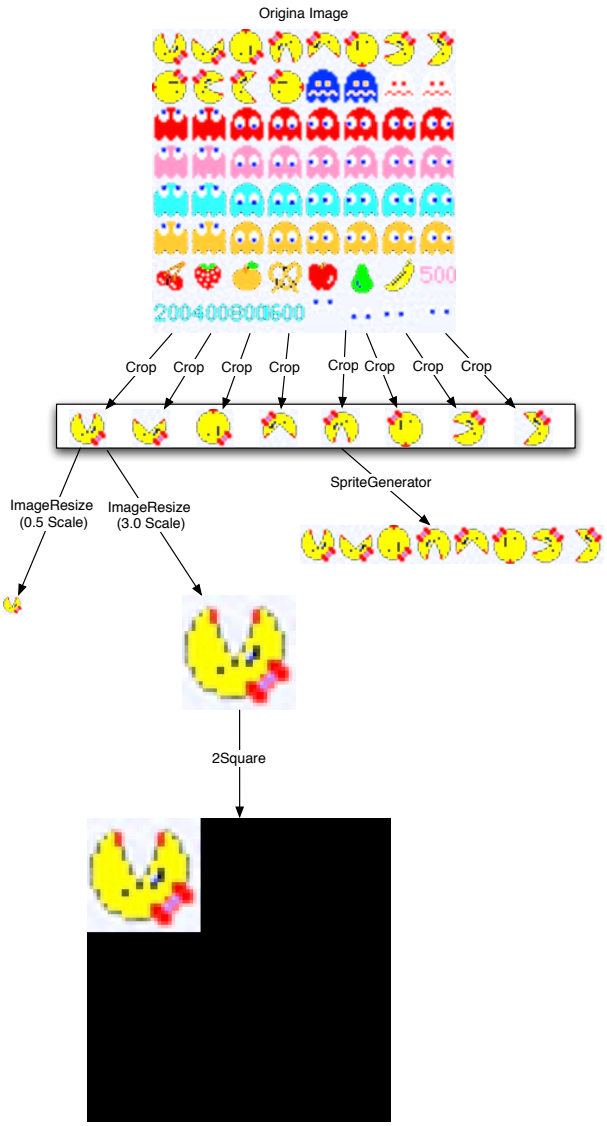


Figure 5: *A Diagram with the main adaptation made by the CAA.*

Moreover, devices can have connection constraints, like availability or cost. The presented architecture also provides a version control system and a cache system to allow the use of offline data and keep data communication to a minimum.

This architecture will be further explored by adding methods for content adaptation, such as needed for 3D models and scenes. Future work also include the evaluation of the proposed architecture using performance evaluation methods methods [Eeckhout 2010] such as the ATAM [Lionberger and Zhang 2007] or SARA [Kruchten et al. 2002].

Acknowledgements

The authors would like to thank the reviewers for their time to help improve this paper. Special thanks to Hermano Cintra for his help on the final revision.

References

AGBOMA, F., AND LIOTTA, A. 2010. Quality of experience management in mobile content delivery systems. *Telecommunication Systems 49*, 1, 85–98.

[4] http://www.kesiev.com/akihabara/

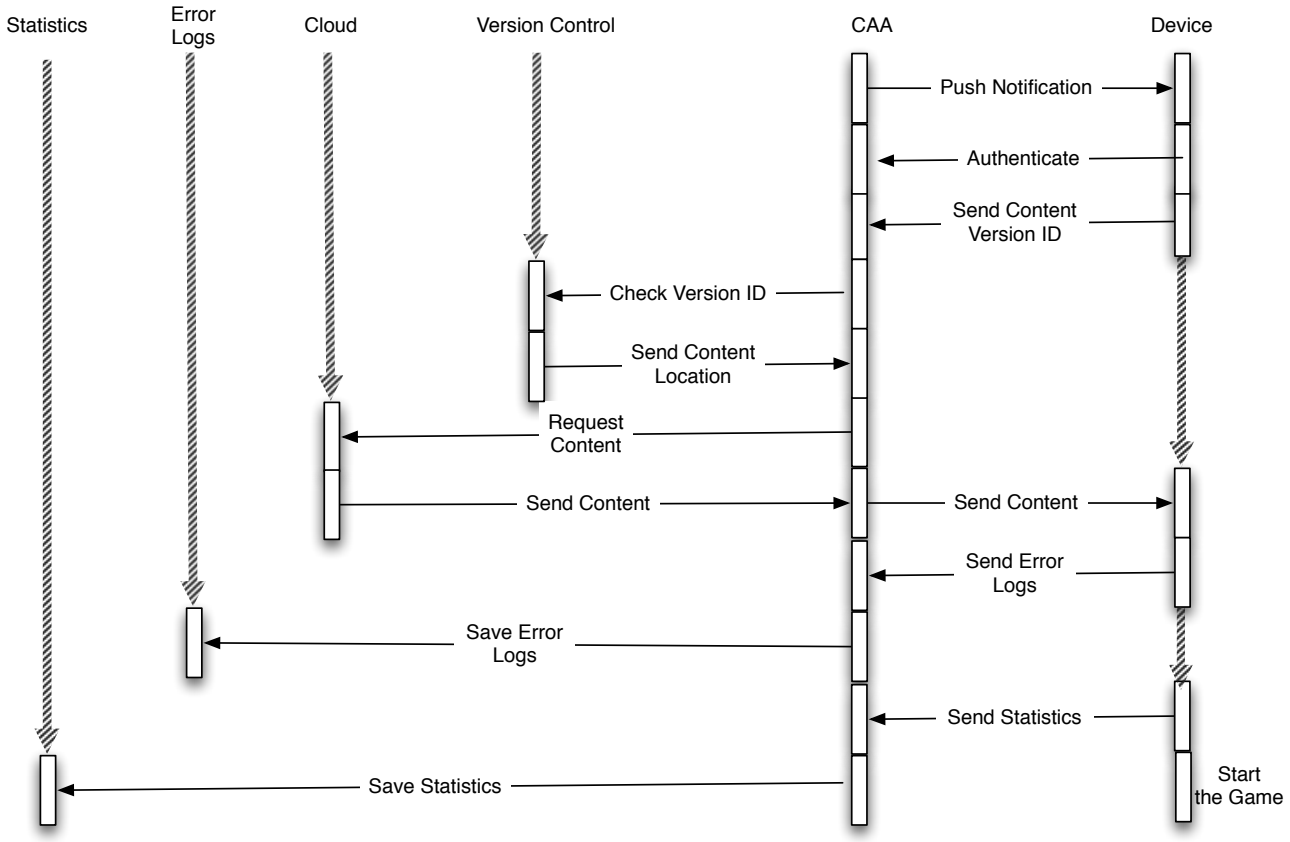
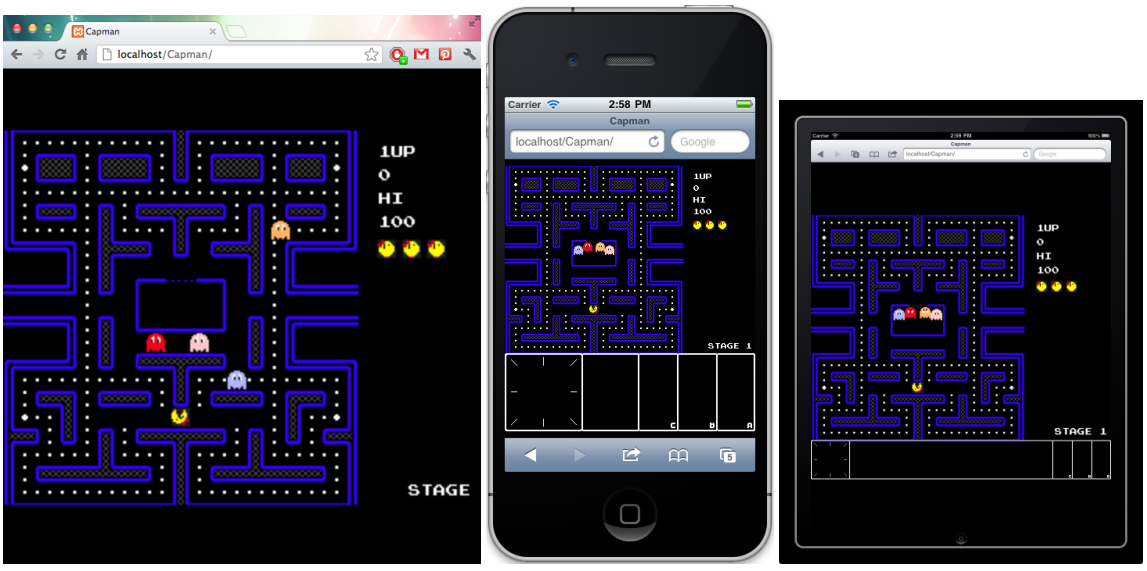


Figure 4: *Overview of the Execution on the Client.*



(a) Game on the Web (b) Game on a smartphone (c) Game on a tablet

Figure 6: *The CMS Adapter Architecture in action adapting html content and image.*

CARVALHO, D., AND TRINTA, F. 2009. Content adaptation for multiplatform applications. In *Proceedings of the XV Brazilian Symposium on Multimedia and the Web*, ACM, New York, NY, USA, WebMedia '09, 41:1–41:4.

CHAARI, T., LAFOREST, F., AND CELENTANO, A. 2007. Adaptation in Context-Aware Pervasive Information Systems: The SECAS Project. *Int. Journal on Pervasive Computing and Communications(IJPCC) 3*, 4 (Dec.), 400–425.

DELICATO, F. C., SANTOS, I. L. A., PIRES, P. F., OLIVEIRA, A. L. S., BATISTA, T., AND PÌRMEZ, L. 2009. Using aspects and dynamic composition to provide context-aware adaptation for mobile applications. In *Proceedings of the 2009 ACM symposium on Applied Computing*, ACM, New York, NY, USA, SAC '09, 456–460.

DIEGO CARVALHO, F. T. 2009. Servico de adaptacao de conteudo para aplicacoes multiplataforma. In *Anais do 3º Simpsio Brasileiro de Componentes, Arquiteturas e Reutilizao de Software (SBCARS 2009)*.

DINIZ, J. R. B., FERRAZ, C. A. G., TRINTA, F. A. M., MELO, H. N., AND SANTOS, L. M. 2008. Avaliao de um servico de gerenciamento de sessao para ambientes de medicina ubqua. In *Proceedings of the 14th Brazilian Symposium on Multimedia and the Web*, ACM, New York, NY, USA, WebMedia '08, 4–11.

EECKHOUT, L. 2010. *Computer Architecture Performance Evaluation Methods*, 1st ed. Morgan & Claypool Publishers.

HILDEBRAND, A., SCHMIDT, T. C., AND ENGELHARDT, M. 2007. Mobile elearning content on demand. *Information Sciences 5*, 2, 94 – 103.

HOLLOWAY, S., AND JULIEN, C. 2010. The case for end-user programming of ubiquitous computing environments. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*.

INTERNATION, M., 2012. Magnolia cms, June.

JOSELLI, M., AND CLUA, E. 2009. grmobile: A framework for touch and accelerometer gesture recognition for mobile games. In *Proceedings of the 2009 VIII Brazilian Symposium on Games and Digital Entertainment*, IEEE Computer Society, Washington, DC, USA, SBGAMES '09, 141–150.

JOSELLI, M., ZAMITH, M., CLUA, E., LEAL-TOLEDO, R., MONTENEGRO, A., VALENTE, L., FEIJO, B., AND PAGLIOSA, P. 2010. An architeture with automatic load balancing for real-time simulation and visualization systems. *JCIS - Journal of Computational Interdisciplinary Sciences*, 207–224.

JOSELLI, M., SILVA JUNIOR, J. R., ZAMITH, M., SOLURI, E., MENDONCA, E., PELEGRINO, M., AND CLUA, E. W. G. 2012. An architecture for game interaction using mobile. In *Games Innovation Conference (IGIC), 2012 IEEE International*, 73–77.

JOSELLI, M., JUNIOR, J. S., VALENTE, L., ZAMITH, M., CLUA, E., FEIJ, B., AND SOLURI, E. 2012. An architecture for mobile games with cloud computing module. In *SBGames 2012 - Trilha de Computao ()*.

JOSELLI, M., SOLURI, E., PASSOS, E., JUNIOR, J. S., ZAMITH, M., AND CLUA, E. 2012. A flocking boids simulation and optimization structure for mobile multicore architectures. In *SBGames 2012 - Trilha de Computao ()*.

KRUCHTEN, P., HILLIARD, R., KAZMAN, R., KOZACZYNSKI, W., OBBINK, H., AND RAN, A. 2002. Workshop on methods and techniques for software architecture review and assessment (sara). In *Proceedings of the 24th International Conference on Software Engineering*, ACM, New York, NY, USA, ICSE '02, 675–675.

LI, W., CHAN, E., CHEN, D., AND LU, S. 2009. Maintaining probabilistic consistency for frequently offline devices in mobile ad hoc networks. In *Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems*, IEEE Computer Society, Washington, DC, USA, ICDCS '09, 215–222.

LIONBERGER, B., AND ZHANG, C. 2007. Atam assistant: a semi-automated tool for the architecture tradeoff analysis method. In *Proceedings of the 11th IASTED International Conference on Software Engineering and Applications*, ACTA Press, Anaheim, CA, USA, SEA '07, 330–335.

MANASHTY, A. R., RAJI, M. R. A., JAHROMI, Z. F., AND RAJABZADEH, A. 2010. Armrayan multimedia mobile cms: a simplified approach towards content-oriented mobile application designing. *Engineering and Technology*, 6.

MFABRIK, 2012. mfabrik web and mobile cms, June.

MOBIMANAGE, 2012. Mobile cms, June.

MORÁN, F., PREDA, M., LAFRUIT, G., VILLEGAS, P., AND BERRETTY, R.-P. 2007. 3d game content distributed adaptation in heterogeneous environments. *EURASIP J. Adv. Signal Process 2007*, 2 (June), 31–31.

NURMINEN, J., WIKMAN, J., KOKKINEN, H., MUILU, P., AND GRNHOLM, M. 2008. Drupal content management system on mobile phone. *2008 5th IEEE Consumer Communications and Networking Conference*, 1228–1229.

RICARDO DA SILVA JUNIOR, J., GONZALEZ CLUA, E. W., MONTENEGRO, A., LAGE, M., DREUX, M. D. A., JOSELLI, M., PAGLIOSA, P. A., AND KURYLA, C. L. 2012. A heterogeneous system based on gpu and multi-core cpu for real-time fluid and rigid body simulation. *International Journal of Computational Fluid Dynamics 26*, 3, 193–204.

SERVICES, C. C. M., 2012. Cellular cms, June.

TRINTA, F., PEDROSA, D., FERRAZ, C., AND RAMALHO, G. 2007. Evaluating a middleware for crossmedia games. In *Proceedings of the 5th international workshop on Middleware for pervasive and ad-hoc computing: held at the ACM/IFIP/USENIX 8th International Middleware Conference*, ACM, New York, NY, USA, MPAC '07, 43–48.

TRINTA, F., PEDROSA, D., FERRAZ, C., AND RAMALHO, G. 2008. Evaluating a middleware for crossmedia games. *Comput. Entertain. 6*, 3 (Nov.), 40:1–40:19.

TUMMALA, H., AND JONES, J. 2005. Developing spatially-aware content management systems for dynamic, location-specific information in mobile environments. *Proceedings of the 3rd ACM international workshop on Wireless mobile applications and services on WLAN hotspots WMASH 05*, 14.

YANNAKAKIS, G. N., AND TOGELIUS, J. 2011. Experience-driven procedural content generation. *IEEE Trans. Affect. Comput. 2*, 3 (July), 147–161.