PUC-Rio - Departamento de Informática

Ginga-NCL com objetos de mídia SSML embutidos

Relatório Técnico: Código Fonte

Rafael Diniz
Matrícula: 1312398

5 de agosto de 2014

Sumário

1 Introdução

1.1 Propósito

Este documento apresenta as modificações feitas no código fonte do Ginga-NCL versão C++ no escopo da disciplina "Projeto Final de Programação", fornecendo material para que o professor possa avaliar o trabalho desenvolvido.

Também estão presentes neste documento o código fonte dos testes sistêmicos e de unidade.

1.2 Público Alvo

Este documento se destina à desenvolvedores de software envolvidos com o Ginga e ao professor da disciplina "Projeto Final de Programação".

2 Código fonte

2.1 SsmlPlayerAdapter.h

```
/******************************************************************************
Este arquivo eh parte da implementacao do ambiente declarativo do middleware
Ginga (Ginga-NCL).

Direitos Autorais Reservados (c) 1989-2014 PUC-Rio/Laboratorio TeleMidia

Este programa eh software livre; voce pode redistribui-lo e/ou modificah-lo sob
os termos da Licenca Publica Geral GNU versao 2 conforme publicada pela Free
Software Foundation.

Este programa eh distribuido na expectativa de que seja util, porem, SEM
NENHUMA GARANTIA; nem mesmo a garantia implicita de COMERCIABILIDADE OU
ADEQUACAO A UMA FINALIDADE ESPECIFICA. Consulte a Licenca Publica Geral do
GNU versao 2 para mais detalhes.

Voce deve ter recebido uma copia da Licenca Publica Geral do GNU versao 2 junto
com este programa; se nao, escreva para a Free Software Foundation, Inc., no
endereco 59 Temple Street, Suite 330, Boston, MA 02111-1307 USA.

Para maiores informacoes:
ncl @ telemidia.puc-rio.br
http://www.ncl.org.br
http://www.ginga.org.br
http://www.telemidia.puc-rio.br
******************************************************************************
This file is part of the declarative environment of middleware Ginga (Ginga-NCL)

Copyright: 1989-2014 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.

This program is free software; you can redistribute it and/or modify it under
the terms of the GNU General Public License version 2 as published by
the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.  See the GNU General Public License version 2 for more
details.
```

```
You should have received a copy of the GNU General Public License version 2
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA

For further information contact:
ncl @ telemidia.puc-rio.br
http://www.ncl.org.br
http://www.ginga.org.br
http://www.telemidia.puc-rio.br
*******************************************************************************/

#ifndef SSMLPLAYERADAPTER_H_
#define SSMLPLAYERADAPTER_H_

#include "../FormatterPlayerAdapter.h"
using namespace ::br::pucrio::telemidia::ginga::ncl::adapters;

#include "system/compat/SystemCompat.h"
using namespace ::br::pucrio::telemidia::ginga::core::system::compat;

#include <string>
using namespace std;

namespace br {
namespace pucrio {
namespace telemidia {
namespace ginga {
namespace ncl {
namespace adapters {
namespace tts {
   class SsmlPlayerAdapter : public FormatterPlayerAdapter {
public:
SsmlPlayerAdapter(IPlayerAdapterManager* manager);
virtual ~SsmlPlayerAdapter(){};

protected:
void createPlayer();
   };
}
}
}
}
}
}
}

#endif /*SSMLPLAYERADAPTER_H_*/
```

2.2 SsmlPlayerAdapter.cpp

```
/*******************************************************************************
Este arquivo eh parte da implementacao do ambiente declarativo do middleware
Ginga (Ginga-NCL).
```

```
#include "gingancl/adapters/tts/SsmlPlayerAdapter.h"

#include "gingancl/adapters/AdaptersComponentSupport.h"

namespace br {
namespace pucrio {
namespace telemidia {
```

```
namespace ginga {
namespace ncl {
namespace adapters {
namespace tts {
SsmlPlayerAdapter::SsmlPlayerAdapter(
IPlayerAdapterManager* manager) :
FormatterPlayerAdapter(manager) {

           typeSet.insert("SsmlPlayerAdapter");
}

void SsmlPlayerAdapter::createPlayer() {
clog << "SsmlPlayerAdapter::createPlayer '" << mrl << "'" << endl;

string paramValue;
CascadingDescriptor* descriptor;

bool hasVisual = false;

#if HAVE_MULTIPROCESS
playerCompName = "PlayerProcess";
player = ((PlayerCreator*)(cm->getObject(playerCompName)))(
myScreen, "SsmlPlayer", &hasVisual);

player->setMrl(mrl, true);

#elif HAVE_COMPONENTS
playerCompName = "SsmlPlayer";
player = ((PlayerCreator*)(cm->getObject(playerCompName)))(
myScreen, mrl.c_str(), &hasVisual);
#else
player = new SsmlPlayer(myScreen, mrl.c_str());
#endif

                clog << "SsmlPlayerAdapter::createPlayer '";
clog << mrl << "' ALL DONE" << endl;


FormatterPlayerAdapter::createPlayer();
}
}
}
}
}
}
}
}

extern "C" ::br::pucrio::telemidia::ginga::ncl::adapters::IPlayerAdapter*
createSsmlAdapter(IPlayerAdapterManager* manager, void* param) {

return new ::br::pucrio::telemidia::ginga::ncl::adapters::tts::
SsmlPlayerAdapter(manager);
```

```
}

extern "C" void destroySsmlAdapter(
::br::pucrio::telemidia::ginga::ncl::adapters::IPlayerAdapter* player) {

delete player;
}
```

2.3 SsmlPlayer.h

```
/*****************************************************************************
Este arquivo eh parte da implementacao do ambiente declarativo do middleware
Ginga (Ginga-NCL).

Direitos Autorais Reservados (c) 1989-2014 PUC-Rio/Laboratorio TeleMidia

Este programa eh software livre; voce pode redistribui-lo e/ou modificah-lo sob
os termos da Licenca Publica Geral GNU versao 2 conforme publicada pela Free
Software Foundation.

Este programa eh distribuido na expectativa de que seja util, porem, SEM
NENHUMA GARANTIA; nem mesmo a garantia implicita de COMERCIABILIDADE OU
ADEQUACAO A UMA FINALIDADE ESPECIFICA. Consulte a Licenca Publica Geral do
GNU versao 2 para mais detalhes.

Voce deve ter recebido uma copia da Licenca Publica Geral do GNU versao 2 junto
com este programa; se nao, escreva para a Free Software Foundation, Inc., no
endereco 59 Temple Street, Suite 330, Boston, MA 02111-1307 USA.

Para maiores informacoes:
ncl @ telemidia.puc-rio.br
http://www.ncl.org.br
http://www.ginga.org.br
http://www.telemidia.puc-rio.br
*****************************************************************************
This file is part of the declarative environment of middleware Ginga (Ginga-NCL)

Copyright: 1989-2014 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.

This program is free software; you can redistribute it and/or modify it under
the terms of the GNU General Public License version 2 as published by
the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.  See the GNU General Public License version 2 for more
details.

You should have received a copy of the GNU General Public License version 2
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA

For further information contact:
ncl @ telemidia.puc-rio.br
```

```
http://www.ncl.org.br
http://www.ginga.org.br
http://www.telemidia.puc-rio.br
*******************************************************************************/

#ifndef SSMLPLAYER_H_
#define SSMLPLAYER_H_

#include "mb/interface/IImageProvider.h"
#include "mb/interface/IWindow.h"
using namespace ::br::pucrio::telemidia::ginga::core::mb;

#include "system/fs/IGingaLocatorFactory.h"
using namespace ::br::pucrio::telemidia::ginga::core::system::fs;

#include "util/Color.h"
#include "util/functions.h"
using namespace ::br::pucrio::telemidia::util;

#include "system/compat/SystemCompat.h"
using namespace ::br::pucrio::telemidia::ginga::core::system::compat;

#include "tuner/providers/frontends/isdbt/RingBuffer.h"

#include "Player.h"

#include <string>
using namespace std;


namespace br {
namespace pucrio {
namespace telemidia {
namespace ginga {
namespace core {
namespace player {
class SsmlPlayer :
    public Thread,
    public Player {
private:
pthread_mutex_t mutex;
string content;

public:
SsmlPlayer(GingaScreenID screenId, string mrl);
virtual ~SsmlPlayer();

protected:
void setFile(string mrl);

private:
void loadSsml();
```

```
public:
void run();
bool play();
void stop();
void resume();
virtual void setPropertyValue(string name, string value);

bool synth_ended;
bool player_ended;
pthread_mutex_t output_mutex;
pthread_cond_t output_cond;
struct ring_buffer output_buffer;

};
}
}
}
}
}
}

#endif /*SSMLPLAYER_H_*/
```

2.4 SsmlPlayer.cpp

```
/*****************************************************************************
Este arquivo eh parte da implementacao do ambiente declarativo do middleware
Ginga (Ginga-NCL).

Direitos Autorais Reservados (c) 1989-2014 PUC-Rio/Laboratorio TeleMidia

Este programa eh software livre; voce pode redistribui-lo e/ou modificah-lo sob
os termos da Licenca Publica Geral GNU versao 2 conforme publicada pela Free
Software Foundation.

Este programa eh distribuido na expectativa de que seja util, porem, SEM
NENHUMA GARANTIA; nem mesmo a garantia implicita de COMERCIABILIDADE OU
ADEQUACAO A UMA FINALIDADE ESPECIFICA. Consulte a Licenca Publica Geral do
GNU versao 2 para mais detalhes.

Voce deve ter recebido uma copia da Licenca Publica Geral do GNU versao 2 junto
com este programa; se nao, escreva para a Free Software Foundation, Inc., no
endereco 59 Temple Street, Suite 330, Boston, MA 02111-1307 USA.

Para maiores informacoes:
ncl @ telemidia.puc-rio.br
http://www.ncl.org.br
http://www.ginga.org.br
http://www.telemidia.puc-rio.br
*****************************************************************************
This file is part of the declarative environment of middleware Ginga (Ginga-NCL)

Copyright: 1989-2014 PUC-RIO/LABORATORIO TELEMIDIA, All Rights Reserved.
```

```
This program is free software; you can redistribute it and/or modify it under
the terms of the GNU General Public License version 2 as published by
the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.  See the GNU General Public License version 2 for more
details.

You should have received a copy of the GNU General Public License version 2
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA

For further information contact:
ncl @ telemidia.puc-rio.br
http://www.ncl.org.br
http://www.ginga.org.br
http://www.telemidia.puc-rio.br
*******************************************************************************/

#include <cstdlib>

#include "player/SsmlPlayer.h"

#include "player/PlayersComponentSupport.h"

// TODO: fix the build system to correctly identify this prerequisite.
#define HAVE_CEREVOICE 1

#if HAVE_CEREVOICE

// size of the max read by the voice syntetizer
#define MAX_READ 100000

#include <cerevoice_eng.h>

// We are compatible with other Ginga audio players, which use SDL2
#include <SDL.h>

/* Callback function

   The callback function is fired for every phrase returned by the
   synthesiser.  This simple callback cues the audio buffer in the
   player object.

   To track the changes in status, we pass a pointer to a user data
   structure containing the audio player and the last audio buffer.
*/

typedef struct user_data {
    void *player;
    void *userdata;
    /* Add other user-specific settings here */
```

```
} user_data;

// Callback method which delivers the synthetized audio samples.
static void channel_callback(CPRC_abuf * abuf, void * userdata) {
    /* Transcription buffer, holds information on phone timings,
       markers etc.
     */
    const CPRC_abuf_trans * trans;
    const char * name;
    float start, end;
    int i;
    int wav_mk;
    int wav_done;
    void *addr;

    /* Used for processing when a min/max phone range has been set */
    wav_mk = CPRC_abuf_wav_mk(abuf);
    wav_done = CPRC_abuf_wav_done(abuf);
    printf("INFO: wav_mk %i, wav_done %i\n", wav_mk, wav_done);

    if (wav_mk < 0) wav_mk = 0;
    if (wav_done < 0) wav_done = 0;

    /* Process the transcription buffer items and print information. */
    for(i = 0; i < CPRC_abuf_trans_sz(abuf); i++) {
trans = CPRC_abuf_get_trans(abuf, i);
start = CPRC_abuf_trans_start(trans);
end = CPRC_abuf_trans_end(trans);
name = CPRC_abuf_trans_name(trans);
if (CPRC_abuf_trans_type(trans) == CPRC_ABUF_TRANS_PHONE) {
    printf("INFO: phoneme: %.3f %.3f %s\n", start, end, name);
} else if (CPRC_abuf_trans_type(trans) == CPRC_ABUF_TRANS_WORD) {
    printf("INFO: word: %.3f %.3f %s\n", start, end, name);
} else if (CPRC_abuf_trans_type(trans) == CPRC_ABUF_TRANS_MARK) {
    printf("INFO: marker: %.3f %.3f %s\n", start, end, name);
} else if (CPRC_abuf_trans_type(trans) == CPRC_ABUF_TRANS_ERROR) {
    printf("ERROR: could not retrieve transcription at '%d'", i);
}
    }

    // audio buffer chunk size is "wav_done - wav_mk"
    short *audio_buffer = CPRC_abuf_wav_data(abuf) + wav_mk;
    int bytes_read = wav_done - wav_mk;

    SsmlPlayer *obj = (SsmlPlayer *) ((user_data *)userdata)->userdata;

try_again_write:                            // * 2 because each audio sample has 2 bytes
    if (ring_buffer_count_free_bytes (&(obj->output_buffer)) >= (bytes_read * 2))
    {
pthread_mutex_lock(&(obj->output_mutex));
addr = ring_buffer_write_address (&(obj->output_buffer));
memcpy(addr, audio_buffer, (bytes_read * 2));
ring_buffer_write_advance(&(obj->output_buffer), (bytes_read * 2));
```

```
pthread_cond_signal(&(obj->output_cond));
pthread_mutex_unlock(&(obj->output_mutex));
    }
    else
    {
fprintf(stderr, "Input buffer full, this is not an error, but it's not good.");
pthread_mutex_lock(&(obj->output_mutex));
pthread_cond_wait(&(obj->output_cond), &(obj->output_mutex));
pthread_mutex_unlock(&(obj->output_mutex));
goto try_again_write;
    }

}

// Callback method which plays the audio samples.
static void sdl_audio_callback(void *udata, Uint8 *stream, int len)
{
    void *addr;
    Uint8 buffer[4096];
    int new_len = 0;

    // make sure the initial buffer is silence.
    SDL_memset(stream, 0, len);

    SsmlPlayer *obj = (SsmlPlayer *)udata;

    if (obj->synth_ended == false && ring_buffer_count_bytes(&(obj->output_buffer)) == 0)
return;

    if (obj->synth_ended == true && ring_buffer_count_bytes(&(obj->output_buffer)) == 0)
    {
obj->player_ended = true;
return;
    }

    if (ring_buffer_count_bytes(&(obj->output_buffer)) < len)
    {
pthread_mutex_lock(&(obj->output_mutex));
new_len = ring_buffer_count_bytes(&(obj->output_buffer));
addr = ring_buffer_read_address(&(obj->output_buffer));
memcpy(buffer, addr, new_len);
ring_buffer_read_advance(&(obj->output_buffer), new_len);
pthread_cond_signal(&(obj->output_cond));
pthread_mutex_unlock(&(obj->output_mutex));

    }
    else if (ring_buffer_count_bytes(&(obj->output_buffer)) >= len)
    {
pthread_mutex_lock(&(obj->output_mutex));
new_len = len;
addr = ring_buffer_read_address(&(obj->output_buffer));
memcpy(buffer, addr, new_len);
ring_buffer_read_advance(&(obj->output_buffer), new_len);
```

```
pthread_cond_signal(&(obj->output_cond));
pthread_mutex_unlock(&(obj->output_mutex));
    }

    // mix our audio against the silence
    SDL_MixAudio(stream, buffer, new_len, SDL_MIX_MAXVOLUME);

}

#endif

namespace br {
namespace pucrio {
namespace telemidia {
namespace ginga {
namespace core {
namespace player {

    SsmlPlayer::SsmlPlayer(GingaScreenID screenId, string mrl) :
        Thread(), Player(screenId, mrl) {

        Thread::mutexInit(&mutex, NULL);

        pthread_mutex_init(&output_mutex, NULL);
        pthread_cond_init(&output_cond, NULL);
        ring_buffer_create(&output_buffer, 28);

    }

    SsmlPlayer::~SsmlPlayer() {
        synth_ended = true;
        player_ended = true;

        ring_buffer_free (&output_buffer);
        pthread_mutex_destroy(&output_mutex);
        pthread_cond_destroy(&output_cond);

        Thread::mutexLock(&mutex);
        Thread::mutexUnlock(&mutex);
        Thread::mutexDestroy(&mutex);


    }

    void SsmlPlayer::setFile(string mrl) {
        clog << "SsmlPlayer::setFile!! " << endl;

        if (mrl == "" || !fileExists(mrl)) {
            clog << "SsmlPlayer::setFile Warning! File not found: '";
            clog << mrl << "'" << endl;
            return;
        }
```

```
        if (mrl.length() > 5) {
            string fileType;

            this->mrl = mrl;
            fileType = this->mrl.substr(this->mrl.length() - 5, 5);
            if (fileType != ".ssml") {
                clog << "SsmlPlayer::loadFile Warning! Unknown file ";
                clog << "type for: '" << this->mrl << "'" << endl;
            }

        } else {
            clog << "SsmlPlayer::loadFile Warning! Unknown extension ";
            clog << "type for: '" << mrl << "'" << endl;
        }
    }

    // This method is the most important one. It sets up the audio synthesizer, the
    // output audio device, reads the input SSML file and calls the apropriate
    // methods to perform the audio synthesis and playback.
    void SsmlPlayer::loadSsml() {
#if HAVE_CEREVOICE
        ifstream fis;

        clog << "SsmlPlayer::loadSsml!! " << endl;

        Thread::mutexLock(&mutex);

        fis.open((this->mrl).c_str(), ifstream::in);

        if (!fis.is_open() && (mrl != "" || content == "")) {
            clog << "SsmlPlayer::loadFile Warning! can't open input ";
            clog << "file: '" << this->mrl << "'" << endl;
            Thread::mutexUnlock(&mutex);
            return;
        }
        fis.close();

        // Now the CereProc API
        CPRCEN_engine * eng;
        CPRCEN_channel_handle hc;

        char * voice_file_por = "/tmp/cerevoice_gabriel_3.0.1_22k.voice";
        char * voice_file_eng = "/tmp/cerevoice_heather_3.0.8_22k.voice";
        char * license_file = "/tmp/license.lic";
        char text_buffer[MAX_READ];
        char * ret;
        FILE * text_fp;
        int res, freq;


        /* Create a empty engine object.  The engine maintains the list of
           loaded voices and makes them available to synthesis channels. */
        eng = CPRCEN_engine_new();
```

```
    /* Load a portuguese voice into the engine */
    res = CPRCEN_engine_load_voice(eng, license_file, NULL, voice_file_por,
    CPRC_VOICE_LOAD_EMB_AUDIO);
    if (!res) {
        clog << "SsmlPlayer::loadSsml ERROR: unable to load voice file "
                                << voice_file_por << ", exiting." << endl;
        return;
    }

    /* Load an english voice into the engine */
    res = CPRCEN_engine_load_voice(eng, license_file, NULL, voice_file_eng,
    CPRC_VOICE_LOAD_EMB_AUDIO);
    if (!res) {
        clog << "SsmlPlayer::loadSsml ERROR: unable to load voice file "
                                << voice_file_eng << ", exiting." << endl;
        return;
    }

    // Lets print information about the voices we loaded...
    int num_voices = CPRCEN_engine_get_voice_count(eng);
    for (int i=0; i < num_voices; i++) {
        const char * voicename = CPRCEN_engine_get_voice_info(eng, i, "VOICE_NAME");
        clog << "SsmlPlayer::loadSsml Voice name: " << i << " is " << voicename << endl;
    }

    /* Open a synthesis channel based in the user selected language */
    hc = CPRCEN_engine_open_channel(eng, "pt", "br", "", "");
    // hc = CPRCEN_engine_open_channel(eng, "en", "gb", "", "");

    // This sets the default phrase-by-phrase output
    CPRCEN_channel_set_phone_min_max(eng, hc, 0, 0);

    freq = atoi(CPRCEN_channel_get_voice_info(eng, hc, "SAMPLE_RATE"));
    clog << "SsmlPlayer::loadSsml Voice samplerate is: " << freq << endl;

    // void *data = (void *)this;
    user_data data;
    data.player = NULL;
    data.userdata = (void *) this;
    // void *data = NULL;
    SDL_AudioSpec audio_spec;
    SDL_zero(audio_spec);
    audio_spec.freq = freq;
    audio_spec.format = AUDIO_S16LSB;
    audio_spec.channels = 1;
    audio_spec.samples = 4096;
    audio_spec.callback = sdl_audio_callback;
    audio_spec.userdata = (void *) this;

    /* Open the audio device */
sdl_try_again:
    if ( SDL_OpenAudio(&audio_spec, NULL) < 0 ){
```

```
            clog << "SsmlPlayer::loadSsml Couldn't open audio: " << SDL_GetError() << endl;
            SDL_Delay(100);
            goto sdl_try_again;
            return;
        }

        /* Start playing */
        SDL_PauseAudio(0);

        // Sets the callback function for anchors notification
        res = CPRCEN_engine_set_callback(eng, hc, &data, channel_callback);
        if (res) fprintf(stderr, "INFO: callback initialised\n");

        text_fp = fopen(mrl.c_str(), "r");

        /* Synthesise input line-by-line */
        while (!feof(text_fp)) {
            ret = fgets(text_buffer, MAX_READ, text_fp);
            if (!ret) break;
            fprintf(stderr, "INFO: text read '%s'\n", text_buffer);
            /* Synthesise the text buffer - the final argument is 'flush'.
               Do not flush the buffer until all the input is sent.
            */
            CPRCEN_engine_channel_speak(eng, hc, text_buffer, strlen(text_buffer), 0);
        }
        /* Finished processing, flush the buffer with empty input */
        CPRCEN_engine_channel_speak(eng, hc, "", 0, 1);

        /* Clean up. The engine deletion function cleans up all loaded
           voices and open channels */
        CPRCEN_engine_delete(eng);
        fclose(text_fp);

        synth_ended = true;

        // shut everything down
        while (player_ended != true)
            SDL_Delay(100);
        SDL_CloseAudio();

        notifyPlayerListeners(PL_NOTIFY_STOP, "");

        Thread::mutexUnlock(&mutex);

#else // Fallback to espeak if Ginga is compiled without CereVoice support.
      //Espeak is lower quality but works.
        string command;
        clog << "SsmlPlayer::loadSsml!! " << endl;

        ifstream fis;

        Thread::mutexLock(&mutex);
```

```
        fis.open((this->mrl).c_str(), ifstream::in);

        if (!fis.is_open() && (mrl != "" || content == "")) {
            clog << "SsmlPlayer::loadFile Warning! can't open input ";
            clog << "file: '" << this->mrl << "'" << endl;
            Thread::mutexUnlock(&mutex);
            return;
        }

        command = "espeak -s 120 -v mb-br4  -f " + mrl;
        std::system (command.c_str());

        fis.close();

        Thread::mutexUnlock(&mutex);

#endif
    }

    bool SsmlPlayer::play() {
        clog << "SsmlPlayer::play ok" << endl;

        synth_ended = false;
        player_ended = false;

        bool ret = Player::play();
        Thread::startThread();

        return ret;

    }

    void SsmlPlayer::stop() {
        clog << "SsmlPlayer::stop ok" << endl;

        synth_ended = true;
        player_ended = true;

        Player::stop();
    }


    void SsmlPlayer::resume() {
        SsmlPlayer::play();
    }


    void SsmlPlayer::setPropertyValue(string name, string value) {
        Thread::mutexLock(&mutex);

        Player::setPropertyValue(name, value);
        Thread::mutexUnlock(&mutex);
```

```
    }

    void SsmlPlayer::run() {
        clog << "SsmlPlayer::run thread created!" << endl;
        loadSsml();

    }


}
}
}
}
}
}

extern "C" ::br::pucrio::telemidia::ginga::core::player::IPlayer*
createSsmlPlayer(
    GingaScreenID screenId, const char* mrl, bool hasVisual) {

    return (new ::br::pucrio::telemidia::ginga::core::player::
            SsmlPlayer(screenId, (string)mrl));
}

extern "C" void destroySsmlPlayer(
    ::br::pucrio::telemidia::ginga::core::player::IPlayer* p) {

    delete p;
}
```

2.5 teste1.ncl

```
<ncl id="ssmlteste1" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">

<body id="myBod">
<port component="ssml1" id="p0" />
                <media id="ssml1" src="teste1.ssml" type="application/ssml+xml" />
</body>
</ncl>
```

2.6 teste2.ncl

```
<ncl id="ssmlteste2" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">

<body id="myBod">
<port component="ssml2" id="p0" />
                <media id="ssml2" src="teste2.txt" type="application/ssml+xml" />
</body>
</ncl>
```

2.7 teste3.ncl

<ncl id="ssmlteste3"xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile»

<body id="myBod» <port component="ssml3"id="p0"/> <media id="ssml3"src="teste3.ssml"/>
</body> </ncl>

2.8 teste4.ncl

```
<ncl id="ssmlteste4" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
<head>
  <connectorBase>
    <causalConnector id="onBeginStart">
      <simpleCondition role="onBegin"/>
      <simpleAction role="start" max="unbounded" qualifier="par"/>
    </causalConnector>
  </connectorBase>
</head>

<body id="myBod">

  <port component="ssml4" id="p0" />

  <media id="ssml4" src="teste4.ssml">
    <area id="ssmlmark" label="esporte"/>
  </media>

  <media id="pic1" src="pic.jpg">
    <property name="width" value="100%" />
    <property name="height" value="100%" />
  </media>

  <link id="linkSsmlTest" xconnector="onBeginStart">
    <bind component="ssml4" interface="ssmlmark" role="onBegin"/>
    <bind component="pic1" role="start"/>
  </link>


</body>
</ncl>
```

2.9 teste5.ncl

```
<ncl id="ssmlteste5" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">

<body id="myBod">
<port component="ssml5" id="p0" />
                <media id="ssml5" src="teste5.ssml">
                    <property name="soundLevel" value="0.4" />
                    <property name="balanceLevel" value="-0.8" />
                    <property name="trebleLevel" value="1" />
                    <property name="bassLevel" value="0" />
                    <property name="freeze" value="false" />
                </media>
</body>
</ncl>
```

2.10 teste6.ncl

```
<ncl id="ssmlteste6" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">

<body id="myBod">
<port component="ssml6" id="p0" />
                <media id="ssml6" src="teste6.ssml">
                    <property name="talkOverReduction" value="50%" />
                </media>
</body>
</ncl>
```

2.11 teste7.ncl

```
<ncl id="ssmlteste7" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">

<body id="myBod">
<port component="ssml7" id="p0" />
                <media id="ssml7" src="teste7.ssml">
                  <property name="speechLanguage" value="pt-br" />
                  <property name="speechRate" value="60%" />
                </media>
</body>
</ncl>
```

2.12 teste8.ncl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ncl id="focus" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
<head>

  <regionBase>
    <region id="mainDevice" width="100%" height="100%">
      <region id="rS" left="5%" top="5%" width="40%" height="40%" zIndex="7"/>

      <region id="rB1" right="5%" top="10%" width="23%" height="10%" zIndex="7"/>
      <region id="rB2" right="5%" top="30%" width="23%" height="10%" zIndex="7"/>
      <region id="rB3" right="5%" top="50%" width="23%" height="10%" zIndex="7"/>
      <region id="rB4" right="5%" top="70%" width="23%" height="10%" zIndex="7"/>
      <region id="rT1" left="5%" top="55%" width="70%" height="40%" zIndex="7"/>
    </region>
  </regionBase>

  <descriptorBase>
    <descriptor id="dS" region="rS"/>

    <descriptor id="dB1" region="rB1"
     focusIndex="ixB1"
     moveUp="ixB4" moveDown="ixB2"/>

    <descriptor id="dB2" region="rB2"
     focusIndex="ixB2"
     moveUp="ixB1" moveDown="ixB3"/>

    <descriptor id="dB3" region="rB3"
```

```
      focusIndex="ixB3"
      moveUp="ixB2" moveDown="ixB4">
        <descriptorParam name="focused" value="true"/>
    </descriptor>

    <descriptor id="dB4" region="rB4"
      focusIndex="ixB4"
      moveUp="ixB3" moveDown="ixB1" moveLeft="ixT1"/>

    <descriptor id="dT1" region="rT1"
      focusIndex="ixT1"
      moveRight="ixB4"/>

  </descriptorBase>

  <connectorBase>
    <causalConnector id="onBeginStart">
      <simpleCondition role="onBegin"/>
      <simpleAction role="start" max="unbounded" qualifier="par"/>
    </causalConnector>

    <causalConnector id="onSelectionStop">
      <connectorParam name="value"/>
      <simpleCondition role="onSelection"/>
      <simpleAction role="stop" max="unbounded" qualifier="par"/>
    </causalConnector>

    <causalConnector id="onSelectionStopStart">
      <connectorParam name="value"/>
      <simpleCondition role="onSelection"/>
      <compoundAction operator="seq">
        <simpleAction role="stop" max="unbounded" qualifier="par"/>
        <simpleAction role="start" max="unbounded" qualifier="par"/>
      </compoundAction>
    </causalConnector>

    <causalConnector id="onSelectionSetVar">
      <connectorParam name="value"/>
      <connectorParam name="var"/>
      <simpleCondition role="onSelection"/>
      <simpleAction role="set" max="unbounded" qualifier="par" value="$var"/>
    </causalConnector>

    <causalConnector id="onBeginSetVar">
      <connectorParam name="var"/>
      <simpleCondition role="onBegin"/>
      <simpleAction role="set" value="$var"/>
    </causalConnector>
  </connectorBase>

</head>

<body>
```

```
<port id="entrada" component="s"/>

<media descriptor="dS" id="s" src="logo.png"/>


<!-- menu -->
<media descriptor="dB1" id="b1" src="sobre.txt">
  <property name="fontColor" value="yellow"/>
  <property name="fontStyle" value="text-align:center"/>
  <property name="fontWeight" value="bold"/>
  <property name="fontSize" value="17"/>
</media>

<media descriptor="dB2" id="b2" src="prog.txt">
  <property name="fontColor" value="yellow"/>
  <property name="fontStyle" value="text-align:center"/>
  <property name="fontSize" value="17"/>
</media>

<media descriptor="dB3" id="b3" src="equipe.txt">
  <property name="fontColor" value="yellow"/>
  <property name="fontStyle" value="text-align:center"/>
  <property name="fontSize" value="17"/>
</media>

<media descriptor="dB4" id="b4" src="sair.txt">
  <property name="fontColor" value="red"/>
  <property name="fontStyle" value="text-align:center"/>
  <property name="fontSize" value="17"/>
</media>


<media descriptor="dT1" id="t1" src="text1.txt">
  <property name="fontColor" value="white"/>
  <property name="fontSize" value="15"/>
  <property name="fontStyle" value="text-align:left"/>
</media>

<media descriptor="dT1" id="t2" src="text2.txt">
  <property name="fontColor" value="white"/>
  <property name="fontSize" value="15"/>
  <property name="fontStyle" value="text-align:center"/>
</media>

<media descriptor="dT1" id="t3" src="text3.txt">
  <property name="fontColor" value="white"/>
  <property name="fontSize" value="15"/>
  <property name="fontStyle" value="text-align:left"/>
</media>

<media id="ssml1" src="text1.ssml" />
<media id="ssml2" src="text2.ssml" />
<media id="ssml3" src="text3.ssml" />
```

```
<media id="ssml4" src="sair.ssml"  />

<link id="linkInicio" xconnector="onBeginStart">
  <bind component="s" role="onBegin"/>
  <bind component="b1" role="start"/>
  <bind component="b2" role="start"/>
  <bind component="b3" role="start"/>
  <bind component="b4" role="start"/>
  <bind component="t1" role="start"/>
</link>

<link id="linkInicio2" xconnector="onSelectionStopStart">
  <bind component="b1" role="onSelection"/>
  <bind component="t1" role="start"/>
  <bind component="ssml1" role="start"/>
  <bind component="t2" role="stop"/>
  <bind component="t3" role="stop"/>
  <bind component="ssml2" role="stop"/>
  <bind component="ssml3" role="stop"/>
</link>

<link id="linkInicio3" xconnector="onSelectionStopStart">
  <bind component="b2" role="onSelection"/>
  <bind component="t2" role="start"/>
  <bind component="ssml2" role="start"/>
  <bind component="t1" role="stop"/>
  <bind component="ssml1" role="stop"/>
  <bind component="t3" role="stop"/>
  <bind component="ssml3" role="stop"/>
</link>

<link id="linkInicio4" xconnector="onSelectionStopStart">
  <bind component="b3" role="onSelection"/>
  <bind component="t3" role="start"/>
  <bind component="ssml3" role="start"/>
  <bind component="t1" role="stop"/>
  <bind component="t2" role="stop"/>
  <bind component="ssml1" role="stop"/>
  <bind component="ssml2" role="stop"/>
</link>

<link id="linkTransicaoPraia" xconnector="onSelectionStopStart">
  <bind component="b4" role="onSelection"/>
  <bind component="s" role="stop"/>
  <bind component="b1" role="stop"/>
  <bind component="b2" role="stop"/>
  <bind component="b3" role="stop"/>
  <bind component="b4" role="stop"/>
  <bind component="t1" role="stop"/>
  <bind component="t2" role="stop"/>
  <bind component="t3" role="stop"/>
  <bind component="ssml1" role="stop"/>
  <bind component="ssml2" role="stop"/>
```

```
    <bind component="ssml3" role="stop"/>
    <bind component="ssml4" role="start"/>
  </link>

</body>
</ncl>
```

2.13 TestSsmlPlayer.h

```
#ifndef TestSsmlPlayer_h
#define TestSsmlPlayer_h

#include <iostream>
#include <string>

#include <cppunit/ui/text/TestRunner.h>
#include <cppunit/TestCase.h>
#include <cppunit/TestSuite.h>
#include <cppunit/TestCaller.h>
#include <cppunit/TestRunner.h>
#include <cppunit/TestResult.h>

#include "SsmlPlayer.h"

class SsmlPlayerTestCase : public CppUnit::TestFixture {
public:

    void testConstructor();

    void testCorrectness();

};
#endif
```

2.14 TestSsmlPlayer.cpp

```
#include "TestSsmlPlayer.h"

// method to test the constructor
void SsmlPlayerTestCase::testConstructor() {
    // create the object
    SsmlPlayer player;

    // check that the object is constructed correctly
    CPPUNIT_ASSERT (*player.synth_end == false);
    CPPUNIT_ASSERT (*player.player_end == false);

}


void SsmlPlayerTestCase::testCorrectness() {
    // create the object
    SsmlPlayer player;
```

```
    bool result;
    // test return codes
    result = player.play("teste.ssml");
    CPPUNIT_ASSERT (result == true);

    result = player.stop();
    CPPUNIT_ASSERT (result == true);

}

// the main method
int main (int argc, char* argv[]) {
    CppUnit::TestResult result;
    CppUnit::TestCaller<SsmlPlayerTestCase> test1( "testConstructor",
                                                   &SsmlPlayerTestCase::testConstructor );
    test1.run( &result );


    CppUnit::TestCaller<SsmlPlayerTestCase> test2( "testCorrectness",
                                                   &SsmlPlayerTestCase::testCorrectness );
    test2.run( &result );


    return 0;
}
```