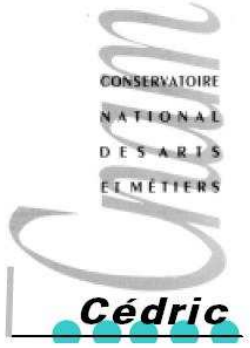
Paris GameDevelopers Conference 08

June 23-24
at Coeur Défense
Paris - FRANCE

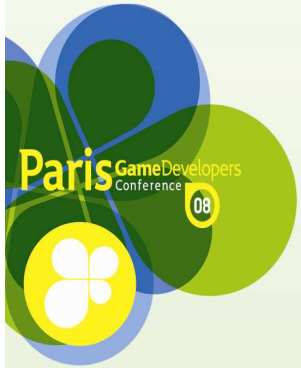


Create a scalable and creative audio environment :

middleware project PLAY ALL

Olivier Veneri
Cédric / Cnam

Yann Planqueel
White Birds Productions



Outline

- State of the Art
- What’s next ?
- PLAY ALL audio framework
- Demo

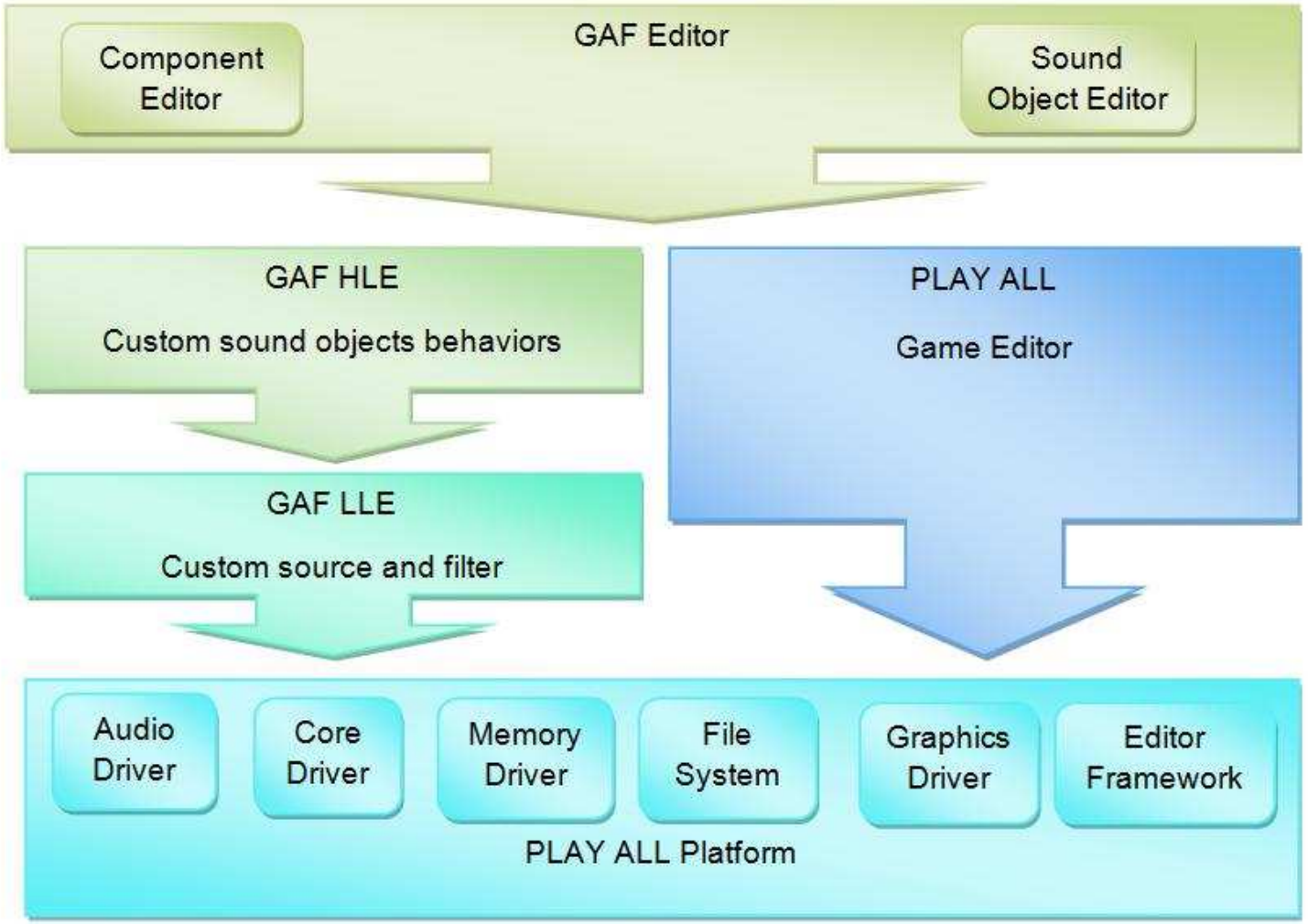
State of the Art

- Sound designer are provided with audio components
  - Sound generator
    - Mainly wave player
    - Possibility to add custom sources
  - Filtering
    - Delay, reverb, low pass…
    - Possibility to add custom filters
  - Dynamic Sound Container
    - Wave file containers with sequencing possibilities
    - Random behavior on some parameters
- Bindings with game states
    - Events
    - Shared variables

What's Next ?

- Procedural Graphics already used in games
  - To create part of graphical asset such as object's texture
  - to generate/adapt character motion
- Procedural Audio is still challenging
  - Sound Synthesis techniques
    - Foley [Doel2001] [Doel2005] [Smith2002]
  - Generative/adaptative music
    - [Malt2000]

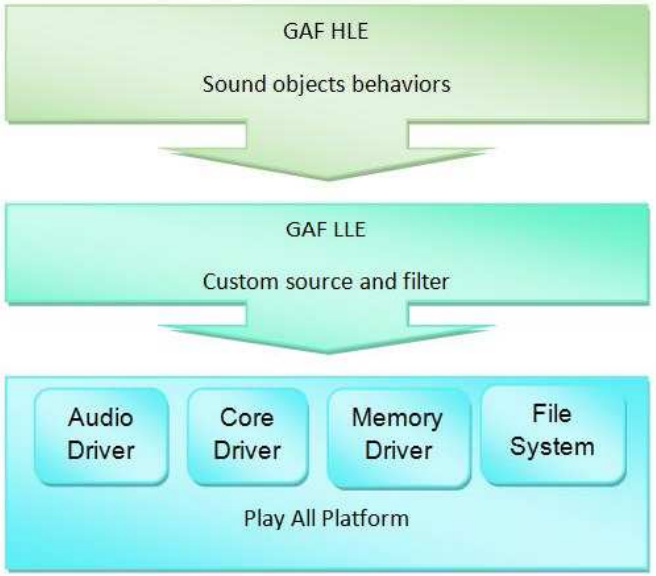
PLAY ALL Audio Framework



Game Audio Framework (GAF) Architecture

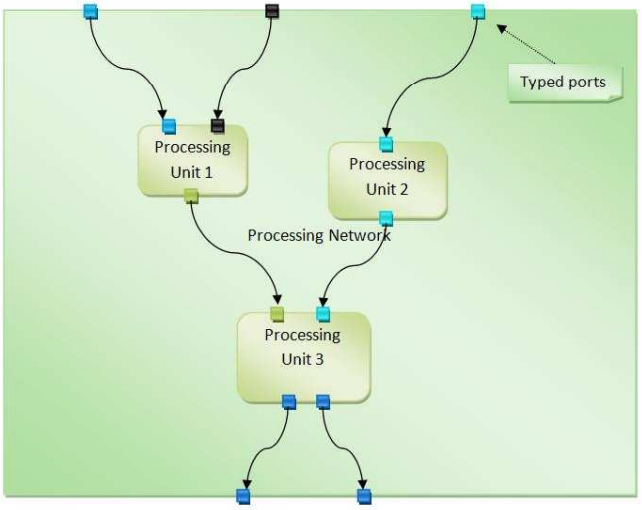
PLAY ALL Audio Framework

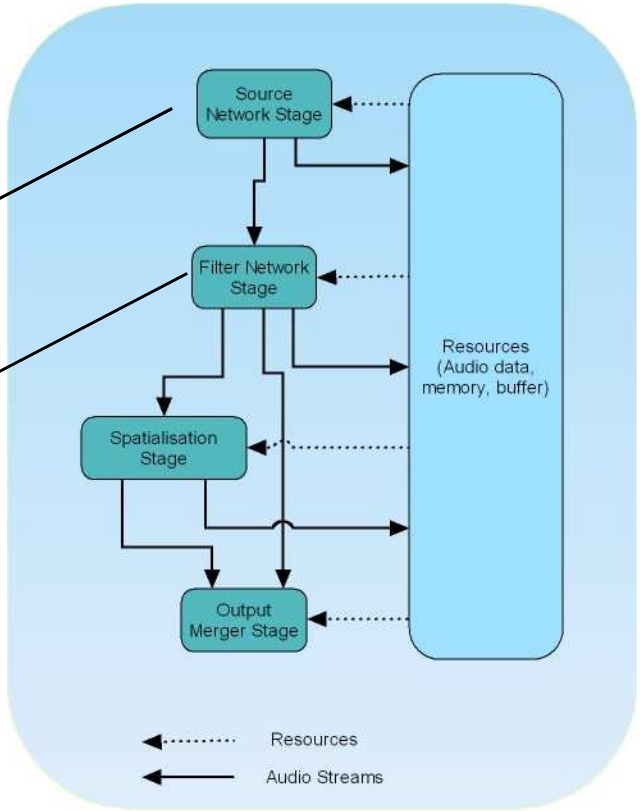
- Based on common audio engine concepts
  - Audio components
  - Game communication
- Built on top of standard sound API with 3D capabilities
  - PLAY ALL sound driver



PLAY ALL Audio Framework

- Defining custom audio components with patches
  - Typed data flow graph
  - Custom sources and filters





- C++ interfaces for custom data types and processing units

PLAY ALL Audio Framework

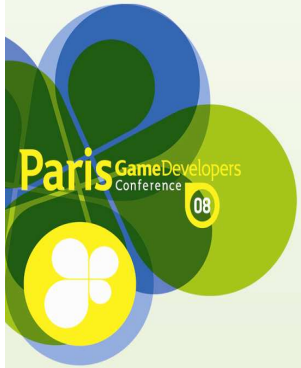
- How to make procedural music ?
- Component approach for sound behaviors
- Scripting language for defining Sound Objects behaviors
  - Include game communication semantic (events and shared variables)
  - Include time manipulation and duration type [Wang2003]
  - Pipeline objects manipulation.

PLAY ALL Audio Framework

- Carefully design for efficiency
  - Compiled, Statically typed => no memory allocation during runtime
  - Glue code only
    - Script can handle C++ object's methods call
    - Reflection mechanism

```
//Custom sound object definition
soundobject WaveLoop {

        source gTock = "FileReader" ;
        channel gTockChannel;
        shared bool bRoomTwo = true ;
        duration dWaitTime = 2000::ms;

        //Initialization event
        event Init()
        {
                gTock.Load(  "Sounds:/Tock.wav "    );
                gTockChannel.SetSource( gTock );
        }

        //StartSequence event
        event StartSequence()
        {
                gTockChannel.Play();
                while( bRoomTwo )
                {
                        gTock.Play();
                        next = dWaitTime;
                }
        }
}
```

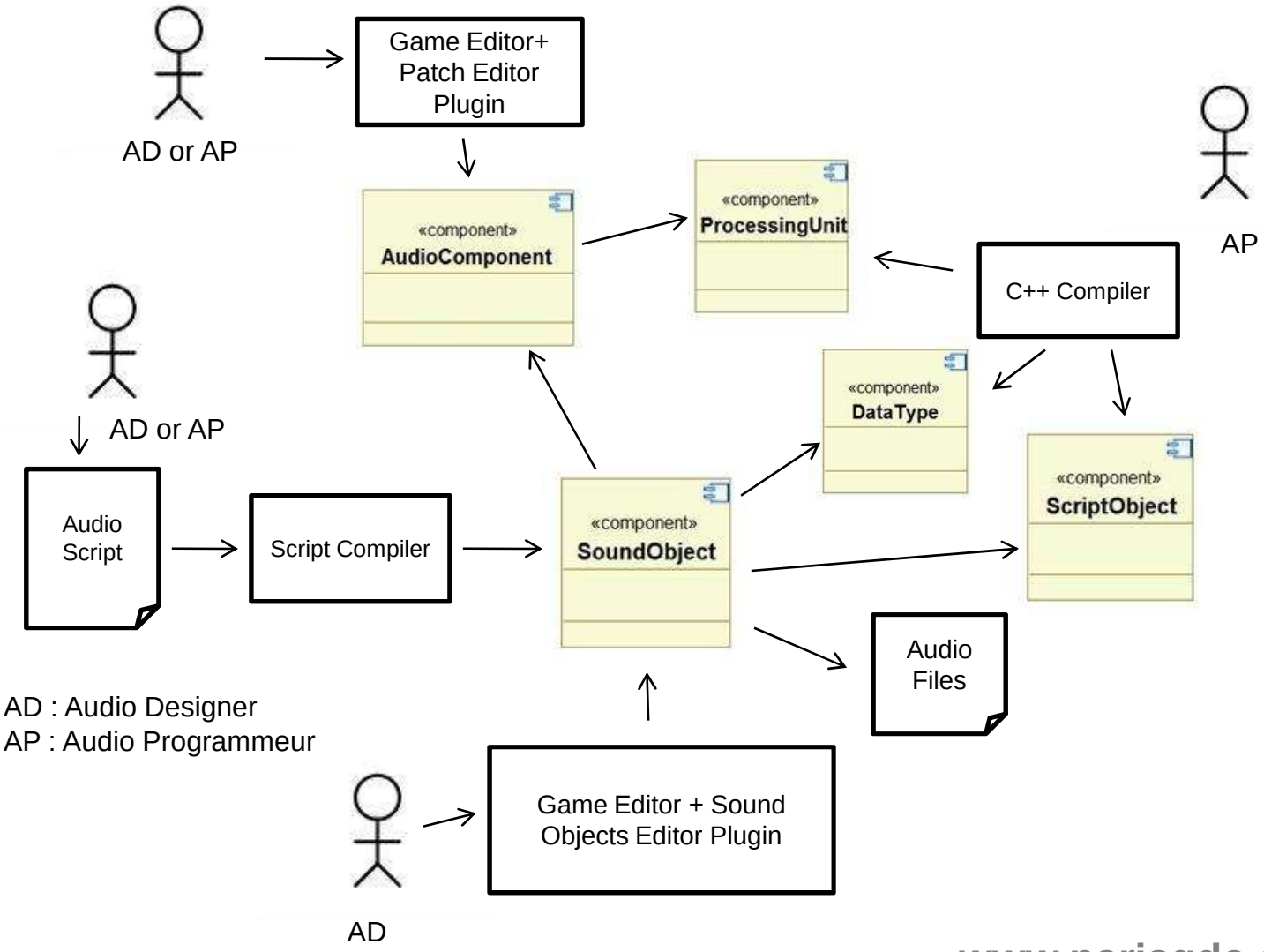
```
SoundObject* gSoundObject = NULL;

if ( SoundObjectManager::CreateSoundObject("WaveLoop.gos", gSoundObject) != PA_OK )
    return PAERR_FAIL;

gSoundObject->CallMethod( "Init", NULL);

gSoundObject->CallMethod( "StartSequence", NULL);

gSoundObject->SetSharedVar( "bRoomTwo", true );
```

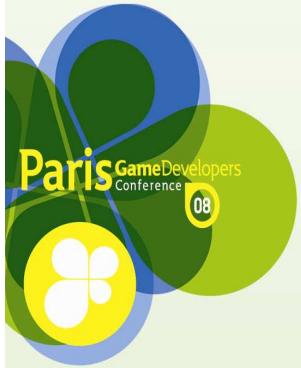


PLAY ALL Audio Framework

- The tools developped for the framework are separated into two levels of abstraction
  - A Max/MSP-like or Reaktor-like editor to build the audio components
  - A mixing-console-like editor to use, organise, tune, the sound objects
- User Defined Interfaces at all Scales
  - The framework allow to define custom interface for sources, filters and sound objects
  - Avoid the common flaw of jungle-like mind-puzzling graphs

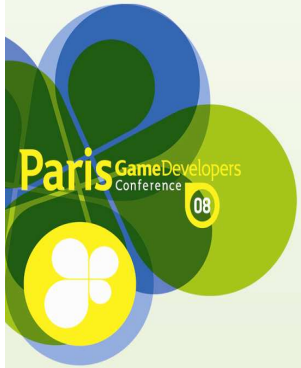
PLAY ALL Audio Framework





PLAY ALL Audio Framework

- The Same and More
  - Typed graphs for audio sources/filters definition
  - Recursive component definition to allow incremental creation process and improve reusability
  - dynamically built interfaces that fit the complexity/level of customization of the created sound pipeline
- User Defined Interfaces at all Scales
  - Avoid the common flaw of jungle-like mind-puzzling graphs
  - Allow and encourage comments and tips about the use of components
  - Allow to tally current standard interfaces



PLAY ALL Audio Framework

- Separation between component definition and interface
  - Optimization
  - User defined organization of the interface

Demo

- We want continuous mapping between game states and musical processes
  - Mixing
  - Note densities
  - Rythmic density
  - Tempo
- Real-Time Music Generation
  - Musical algorithms use game variables

References

- Kees van den Doel, Kry Paul G. & K., P. D.
  Foley Automatic : Physically-based Sound Effects for Interactive Simulation and Animation
  SIGGRAPH 2001, ACM Press, **2001**, 7
- Smyth, T. & III., J.
  The Sounds of the Avian Syrinx-- Are the Really Flute-like?
  Proceedings of DAFX 2002, International Conference on Digital Audio Effects, **2002**
- Wang, G. & Cook, P. R.
  ChucK: A Concurrent, On-the-fly, Audio Programming Language
  ICMC 2003, **2003**
- Physically-based Models for Liquid Sounds
  ACM Transactions on Applied Perception, **2005***, 2*, 534-54
- Mikhail, M.
  Les mathématiques et la Composition Assistée par Ordinateur (Concepts, Outils et Modèles)
  EHESS, **2000**

Questions ?

olivier.veneri@cnam.fr

yplancqueel@playall.fr