Intelligent Sound Bubbles

Dragica Kahlina
sound artist & musician (www.kahlina.com)

GAME DEVELOPERS CONFERENCE®
SAN FRANCISCO, CA
MARCH 17–21, 2014
EXPO DATES: MARCH 19–21
2014

About me

- Musician and sound designer
- was AI coder (C++) for (RTS/RPG-)ga
- physicist

Thanks

- Alex Champandard for ad-hoc mentori
- Guy Somberg for the tips
- Nika Harper for her indie soap box tall

Project : Black Island

- Team : wotokah (wotokah.makegames.ch)
  - Indie -> no time, no money, just ambition
  - 3 People level, code, sound
  - Unity
- Game : Open World, Horror, Oculus Rift
- Published : Halloween 13 after 9 month

We needed

- Dynamic soundscapes
- Have more variety
- Make it easier
  - To input sounds
  - Changing and refining sounds
  - Make dynamic sound sets reusable

Important (for everything)

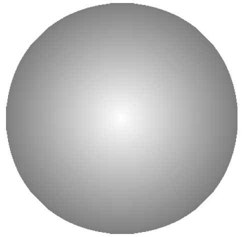
- Ease of use
- Non-blocking multithreading of people

Intelligent Sound Bubbles

packaged sound

&

inbuilt intelligence



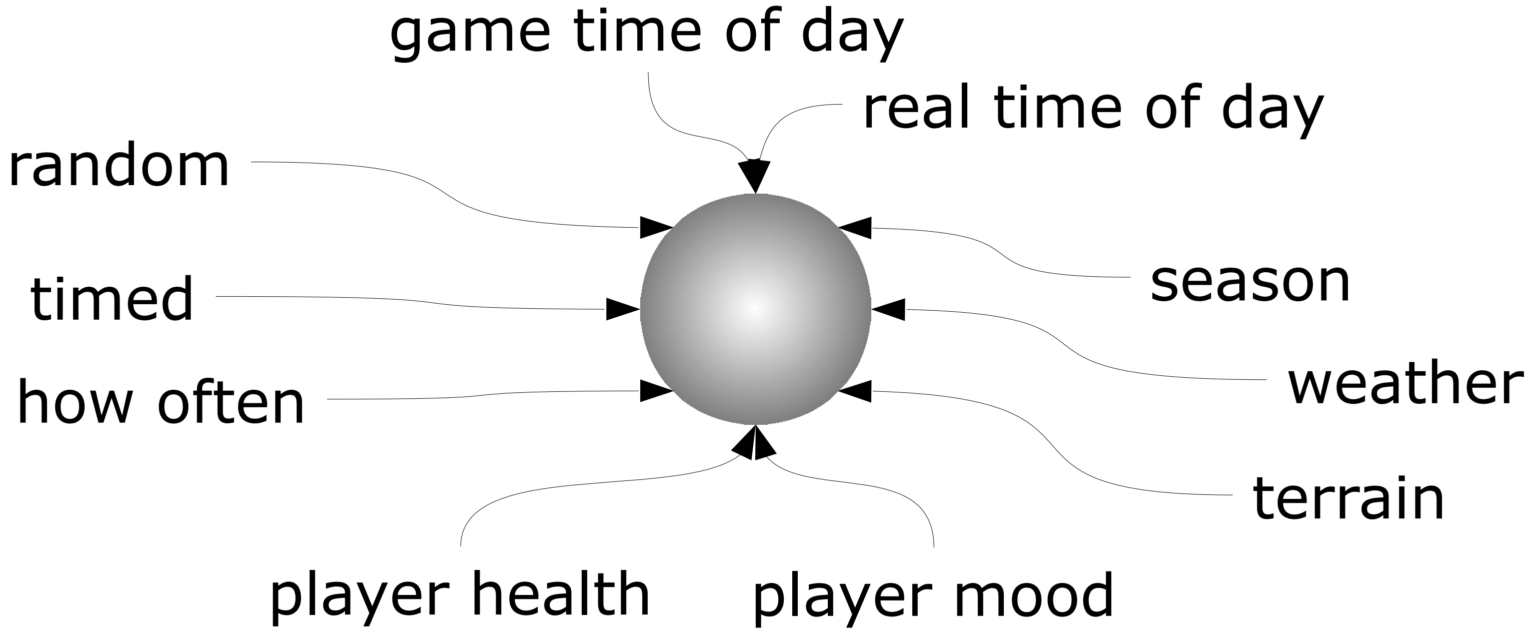
Multi-threading ?

Sound designer
- defines moods
- compiles sound sets
- fills the set
- tweaks parameters from coder (set specific)
- names set & writes description for *level designer*

Level designer
- places sound bubbles in level
- puts name of set in
  if there is no fitting set, gives it a name and hands name + description to *sound designer*
- tweaks parameters from coder (location specific)

Coder
- writes logic
- gives parameters *to sound*
- bubbles (prefab) for *level designer*

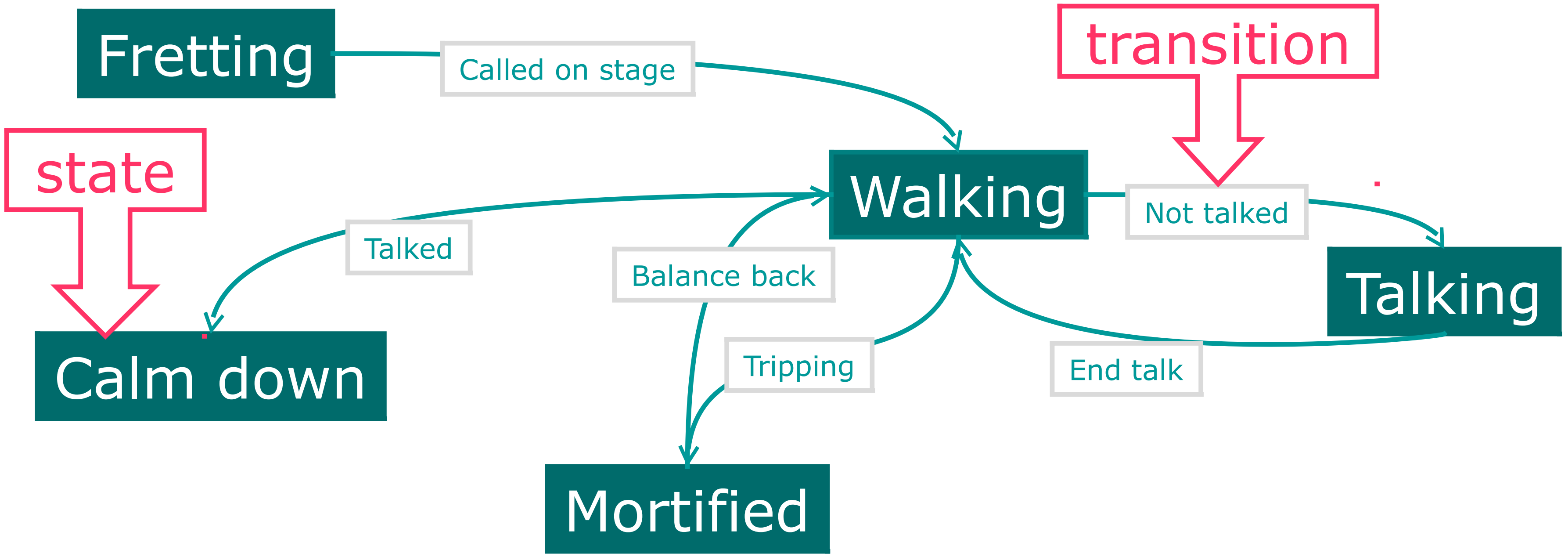
Possible Control Parameters



Use FSM – Finite State Machine

- Mathematical model of computation
- Often used in game AI / enemy behavior
- Different levels of control
- We only use what's needed
- Concept easy to understand

FSM Example – my states



So a Finite State Machine

- Has states, but not infinite many
- Is always in one defined state
- Has conditional transitions
- If a transition becomes true

  -> one state changes to the next

I defined State as …

- Set of different sounds
- With different parameter
  - Time
  - Probability
  - How often
- Intelligence …

State Intelligence

- State switches between sounds
- Switches are based on parameters
- Different functionality -> states don't mix
  - Use different FSM (OOP)
  - Avoid huge monolithic FSM
  - Avoid (for now) FSM of FSMs of …

FSM - Different Levels

FSM Engine
code / library

Controls
- States
- State changes
- Wiring
- General & Reusable

Doesn't have
- State definitions
- Transition definitions

Definitions
code, better data language (XML, JSON, …)

Has
- State definitions
- Transition Definitions
- Intelligence
- Game Specific

Doesn't have
- Data / Parameter

Data
data language (XML, JSON, …)

Has
- Parameters
  - Sound file names
  - Control parameter

Shouldn't be
- Complicated
- In need of a special, homemade editor

Different FSM used

- Local FSM
  - use player trigger to wake up
- Moving FSM (creatures, weather)
  - state moves around
- Music FSM
  - automatic music composition

This leads to Transitions as

- Changes from one sound set to another
- Controled by something called **HAI** (Horror AI) which masterminds horror sounds -> not FSM, avoid overkill
- Based on message system

Intelligent Sound Bubbles

- Dynamic Sound Sets
- Based on one of the base FSM (code)
- Different sets of sounds and parameters defined in XML-File
- Freely distributable in Level
- Can live in an object (player, creature)

In Unity …

- Have prefabs for different FSM
- Distribute in Level or put into an object
- Fill in set ID
- Non-set-parameters, bubble specific

-> level designer can do last 3

How did I get there …

- Disclaimer: I am no Unity or C# guru
- Good FSM needs all the bells and whistles
- Luckily you can find them
  - C++ : Games Programming Gems 2
  - C# / Unity : found one at unitygems.com

Next step (still coding)

- Make children of the FSM base class
- Input states, functionality, transitions
- Try to have this in data language

Data language parser

Used XML, because

- I am used to it
- There are editors that help (didn’t find a good one on Mac for JSON)
- There are XML reader libraries
  - Used TinyXML ported for Unity / C#

Data Structure

```
<Preset>
 <ID>Ghost</ID>
 <States>
  <Ambient>
   <Sound>
    …
   </Sound>
  </Ambient>
  <HORROR1>
   <Sound>
    …
   </Sound>
  </HORROR1>
 </States>
</Preset>
```

Sound Structure

```
<Sound>
    <Filename>Audio/Ghost1</Filename>
    <Timer>100</Timer>
    <Probability>100</Probability>
    <Points>0</Points>
    <Rounds>1</Rounds>
</Sound>
```

Post Mortem

What worked

- Workflow with XML and sound bubbles
- Level designer could just place sound bubbles

What didn't work

- My unfamiliarity with Unity / C#
- Not enough sounds
- Not enough control parameters

Also Music FSM wasn't finished

- My idea
    - Ambitious : short (1-8 bar) precomposed pieces that fit together and change according to some rules
    - Very ambitious : have machine auto-compose and synthesize on the fly

-> working on it, maybe next year

Need sound ?

Dragica Kahlina (www.kahlina.com)
@gluggergames
(and on Facebook, Linkedin, Xing)

wotokah (wotokah.makegames.ch)
@wotokah