

A character with long pink hair, wearing a grey and red outfit, sits on a white, modern-looking bench. The background is a futuristic interior with large windows and circular lights. The text is overlaid in the center.

# **FINAL FANTASY XIII's Motion- Controlled Real-Time Automatic Sound Triggering System**

# Speakers



**Yoshinori Tsuchida**

Technical Director  
Audio Programmer



**Tomohiro Yajima**

Sound Director  
Sound Designer

# **Motion-Controlled Real-Time Automatic Sound Triggering System: Abbreviated as “MASTS”**





# The Value of Sound Effects

## <Physics-Based Acoustic Rendering>

- Add atmospheric depth
- Bring balance to the total of all audio elements
- Allow subtle yet effective expression of character emotions

# Expressing Character Emotions Through Footsteps and Rustling of Clothing

- Expressing emotions through dynamics in weak and strong actions
- Expressing emotions through varying speeds
- Sound production in both upper body close-ups and full-body overhead views
- Sound effects from sneaking and quiet footsteps
- Sound effects from the swinging of arms
- Even determination of whether or not sound effects will occur
- Use in gauging character location and distance

# My Personal Roadmap to Sound Automation - 1



We first conceived of automation during this period when we worked on the complete expression of emotions through footstep and action sound effects to make up for the game's lack of voices.

**Vagrant Story**  
**(PlayStation®/2000)**



# My Personal Roadmap to Sound Automation - 2



We prepared a large number of terrain types and put them in a database so that each terrain could be associated with specific sounds. This, in combination with 3D Panning and Distance Dissipation Algorithms, helped to add realism to the player's experience.

**FINAL FANTASY XI**

**(PlayStation®2, Windows®, Xbox360®/2002～)**

# My Personal Roadmap to Sound Automation - 3

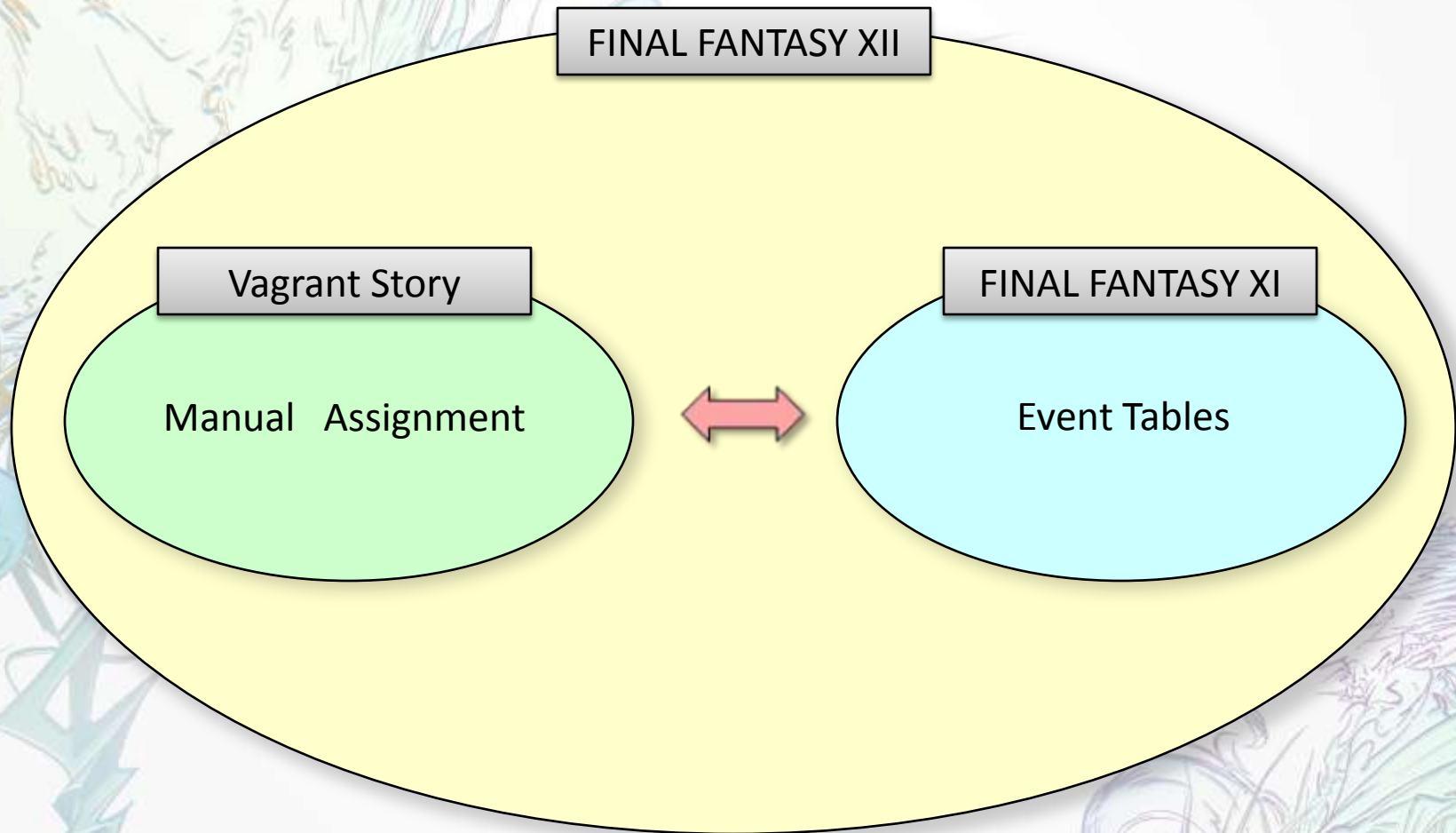


We put all of the know-how we had acquired up to this point into practice and used automation for strong and weak footsteps.

**FINAL FANTASY XII**  
**(PlayStation®2/2006)**



# Benefits and Disadvantages of the Traditional Method



# Reasons for Automation:

## Reason #1

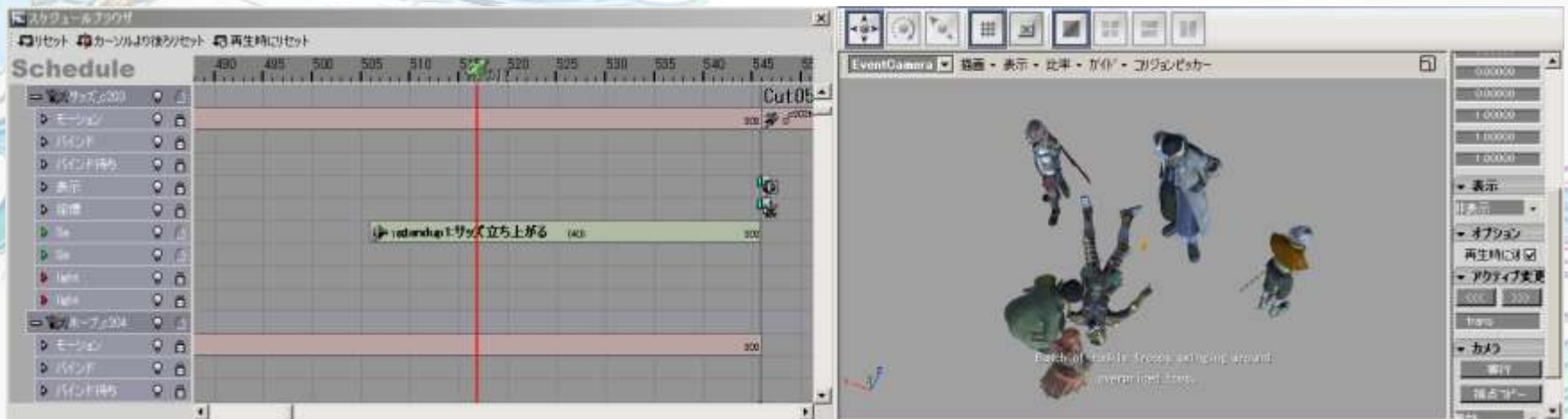
**The work costs of designating each footstep sound individually are huge.**



# Reasons for Automation:

## Reason #2

If an individual motion or an entire scene needs to be changed, the timing of the sounds will need to be minutely adjusted, taking up far too much time.





# Reasons for Automation:

## Reason #3

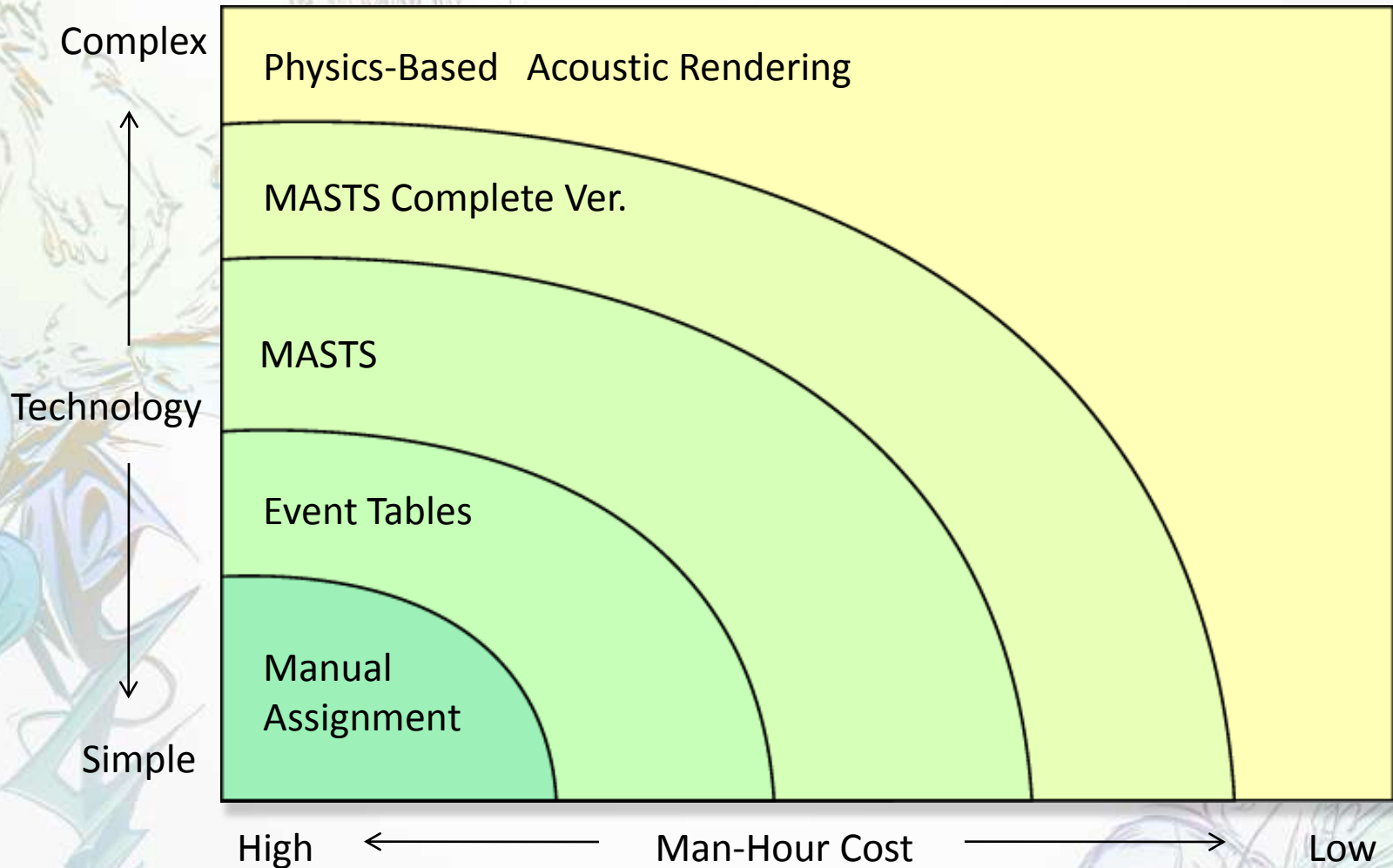
Not good for interactivity.



# FINAL FANTASY XIII – Our New Approach



# FINAL FANTASY XIII – Our New Approach





# Footstep Data in Past Projects

	Number of footstep sounds in table data	Number of cutscene-specific footstep sounds	Dealing with detailed expressions
Vagrant Story	1568	1400	Possible through manual assignment
FF11	1430	Almost none (around 100)	Very difficult due to the wide variety of possible main characters in an MMORPG
FF12	954	8000	Possible through manual assignment
FF13	5200	None	Dealt with through automation

# Moving from Manual Procedures to Automation

## <Original Concept>

- Gyro-sensors inserted in each joint
- Sound effects produced based on speed
- Extent of actions determined based on gyro location and distance
- Sounds determined based on combination of each action

# Challenges of Implementation


- Need for high-level programmers
  - Programmers with experience and expertise in many various fields
- Coordinating with many other non-sound related teams
  - Always a big challenge
- Creating various support tools



# Challenges of Implementation

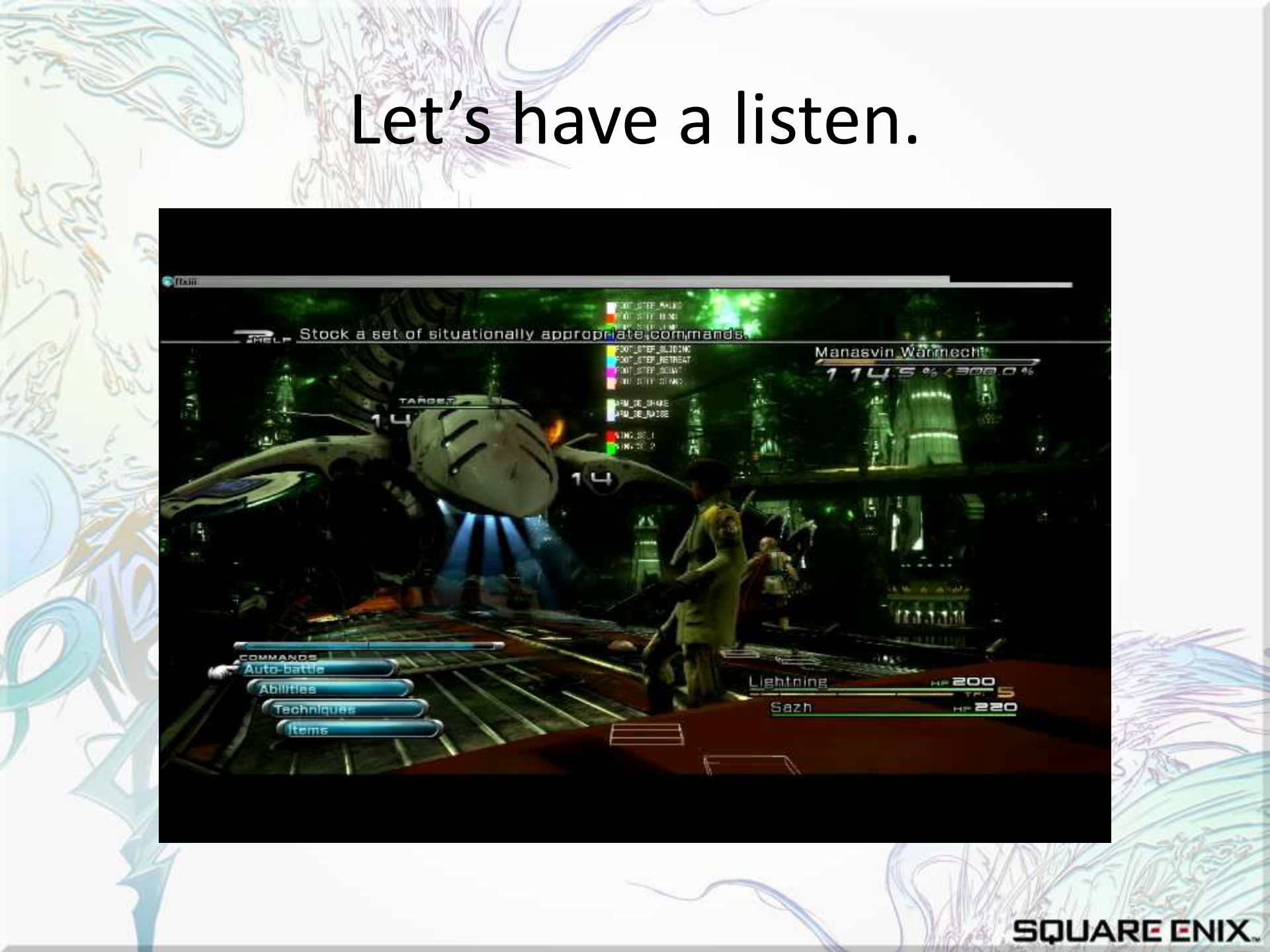
**Sounds like fun! Let's do it!**

# Let's have a listen.




A screenshot from the video game Final Fantasy XIII-2 showing a battle scene. The player's party, consisting of Lightning and Sazh, is fighting a large, white, insect-like enemy named Manasvin Wannecht. The enemy's health is shown as 114.5% / 300.0%. The player's party members have HP bars: Lightning at 200 and Sazh at 220. The interface includes a command menu with options like Auto-battle, Abilities, Techniques, and Items. A text box at the top right says "Stock a set of situationally appropriate commands." The background is a dark, industrial setting with green lighting.

SQUARE ENIX™



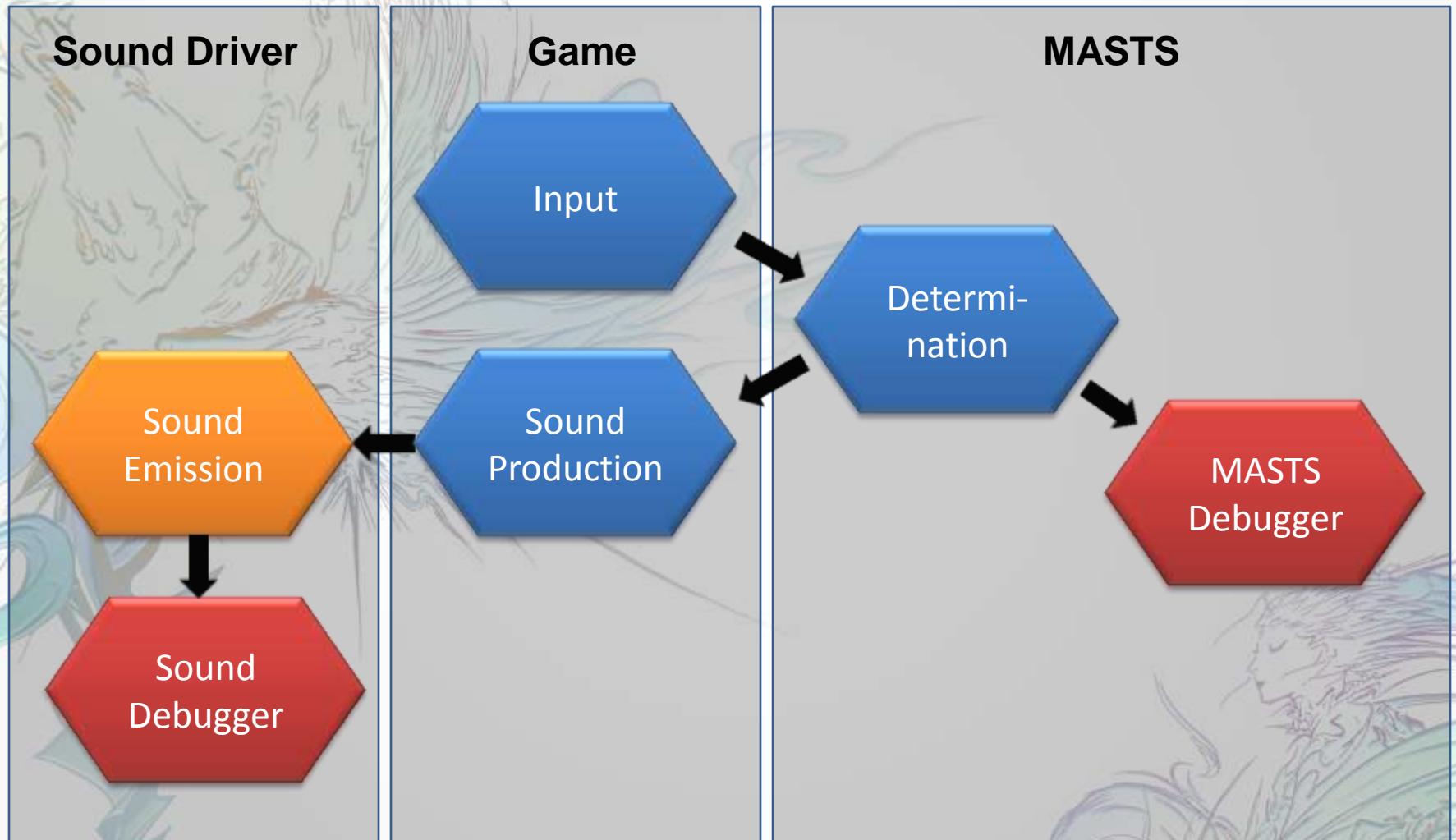
# Let's have a listen.



A screenshot from the video game Final Fantasy XIII-2 showing a battle scene. The player's party, consisting of Lightning and Sazh, is fighting a large, white, insect-like enemy named Manasvin Wannecht. The enemy's health is shown as 114.5% / 300.0%. The player's party members have HP bars: Lightning at 200 and Sazh at 220. The interface includes a command menu with options like Auto-battle, Abilities, Techniques, and Items. A text box at the top right says "Stock a set of situationally appropriate commands." The background is a dark, industrial setting with green lighting.

SQUARE ENIX™

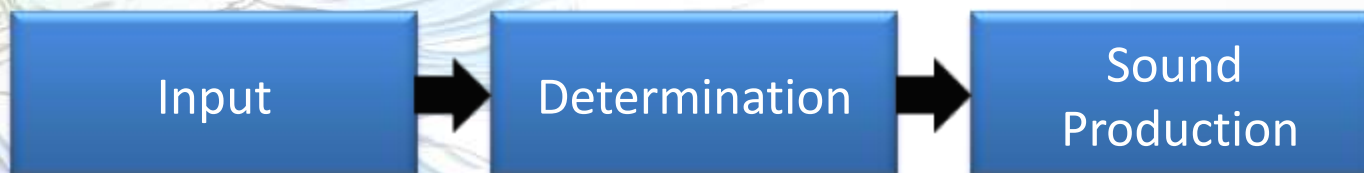
# Overall Data Flow





# The Core of MASTS

- Three phases:

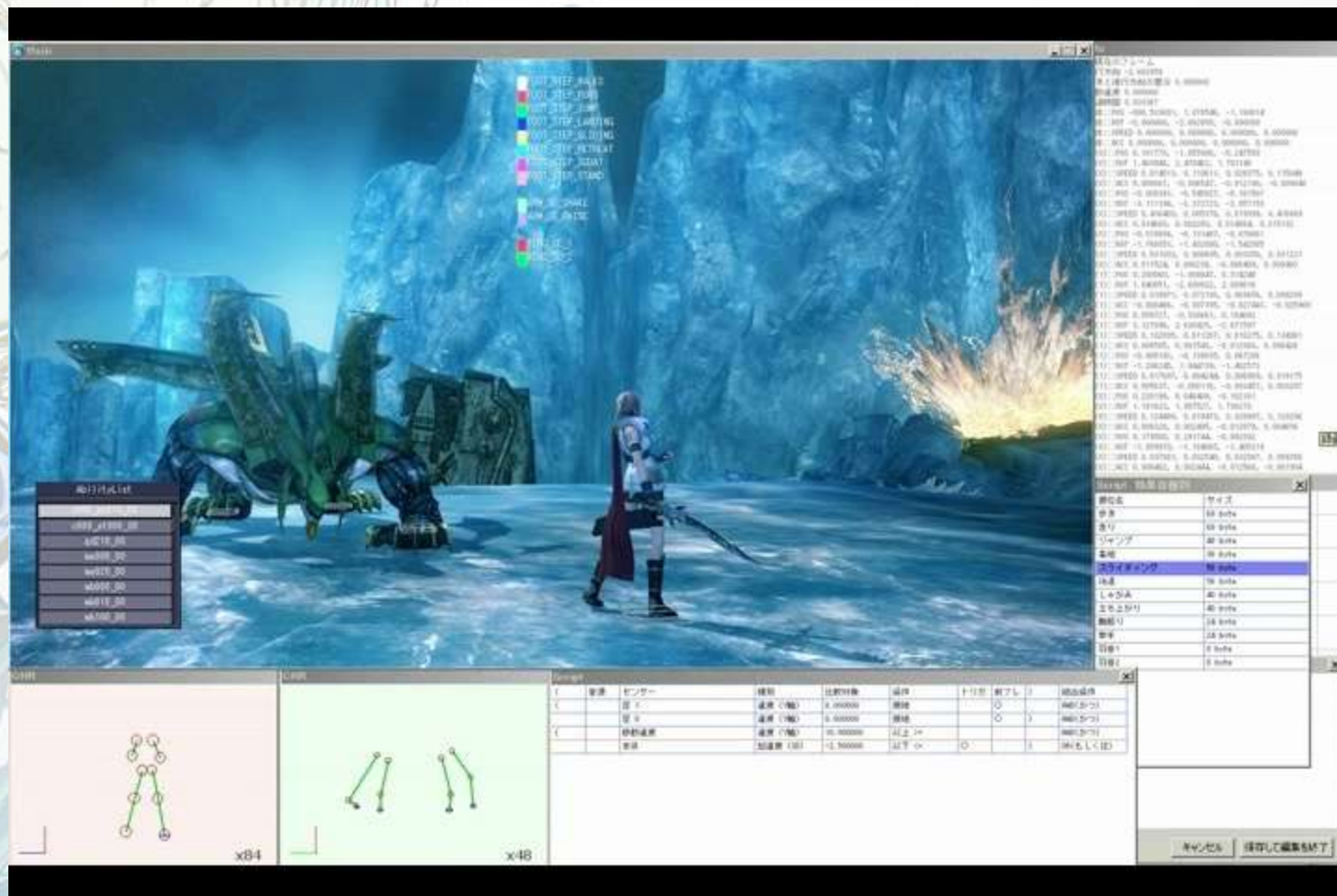


# Input Phase



- Information used in physics calculations:
  - Character's direction, movement speed, etc.
  - Angle and coordinates of character's bones
  - Collision Detection Data
  - Script type (referred to later)
  - Motion type

# Input Phase





# Data Collection and Analysis



- Examples:
  - Character data
    - Location/position, movement direction, orientation, movement speed
    - Motion classification
    - Time elapsed
  - Set of leg bone data —————→
    - Legs, knees, lower back
  - Set of arm bone data
    - Elbows, shoulders
  - Set of wing bone data
    - Shoulders, wing tips

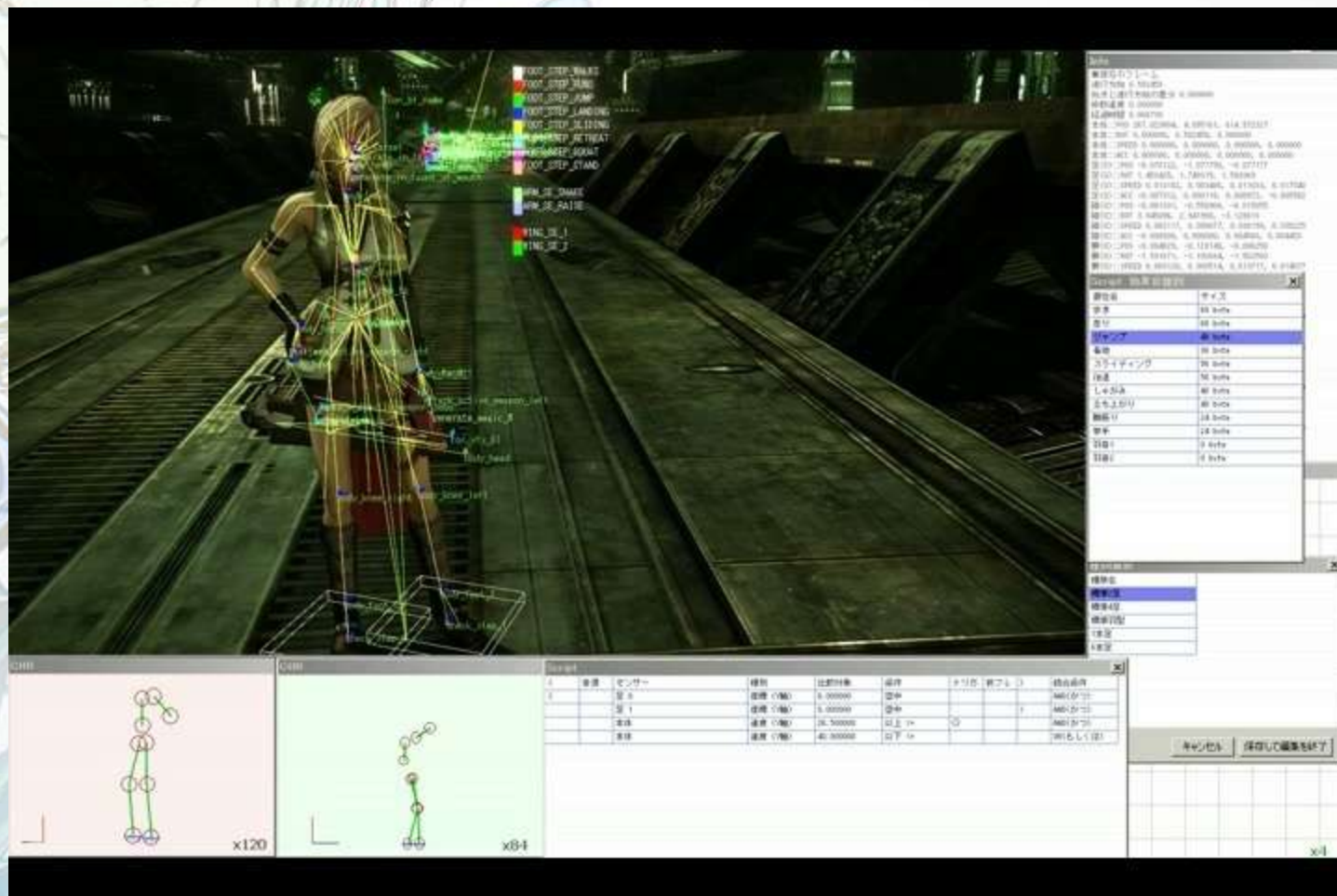
Coordinates  
Angle of rotation  
Speed  
Acceleration

# Determination Phase



- All operations based on inputted data
  - Will a sound be produced or not?
  - What kind of sound will be produced?
    - Landing? Rustling of clothing? Etc.
  - How will the sound be produced?
    - Volume, pitch, etc.

# Determination Phase





# Sound Production Element Chart



- Rustling of Clothing (4 types)
  - Swinging of arms
  - Raising of arms
  - Crouching, standing up
- Footstep Sounds (6 types)
  - Walking (forward and backward)
  - Running
  - Sliding
  - Jumping (ascension, landing)
- Wing Sounds (2 types)
  - 2 types



# Sound Production Phase



- Time for the sound itself! But before that...
  - Character classification, equipment, clothing
  - Terrain type, weather type, location type

“Jumping” sound:

Jumping sound of: main character:  
wearing metal boots: on grass:  
in the rain



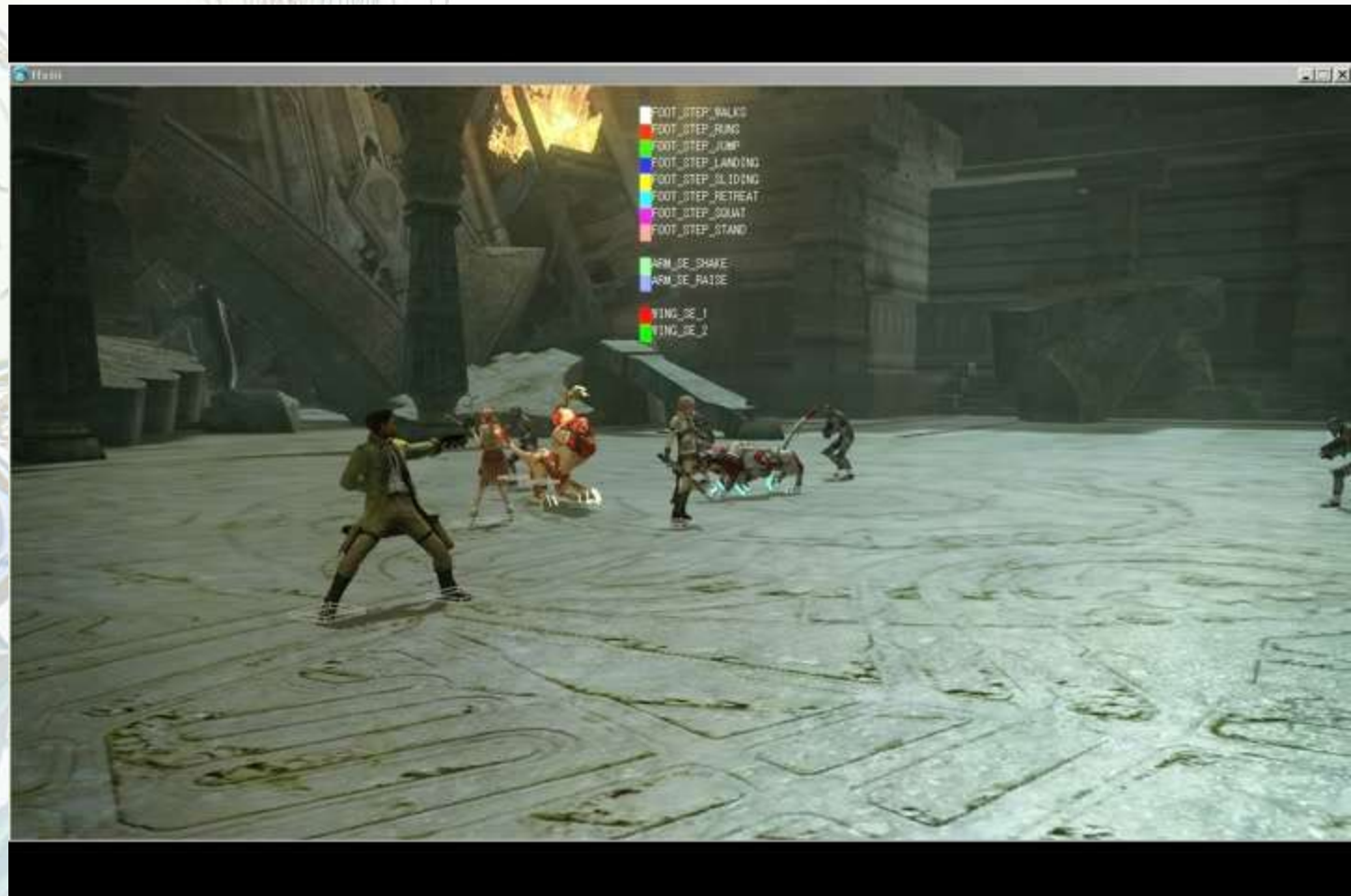
# Sound Production Phase

[illegible][illegible]

A	B	C	D	E	F
10名	説明	TestSoundResourceNameDefault	TestSoundResourceNameDrySolo	TestSoundResourceNameDemo	TestSoundResourceNameGroup
	soundboxにアップ	audioにアップ			
	コンバート				
1					
2	attreff_default	デフォルトセット	fl_npc00_010	fl_npc00_010	fl_npc00_010
3	attreff_1gls	セクト1の周	fl_1light_010	fl_1light_010	fl_1light_010
4	attreff_1gls2	セクト2の周	fl_1gls2_010	fl_1gls2_010	fl_1gls2_010
5	attreff_0ash	セクト0の周	fl_0ash_010	fl_0ash_010	fl_0ash_010
6	attreff_vani	セクト0の周	fl_vani_010	fl_vani_010	fl_vani_010
7	attreff_hope	セクト0の周	fl_hope_010	fl_hope_010	fl_hope_010
8	attreff_hang	セクト0の周	fl_hang_010	fl_hang_010	fl_hang_010
9	attreff_low	セクト0の周	fl_low_010	fl_low_010	fl_low_010
10	attreff_gdot	セクト0の周	fl_gdot_010	fl_gdot_010	fl_gdot_010
11	attreff_lebr	セクト0の周	fl_lebr_010	fl_lebr_010	fl_lebr_010
12	attreff_npc	NPCの周	fl_npc00_010	fl_npc00_010	fl_npc00_010
13	attreff_0dm	オーディン1の周	fl_0dm_010	fl_0dm_010	fl_0dm_010
14	attreff_0dm2	オーディン2の周	fl_0dm2_010	fl_0dm2_010	fl_0dm2_010
15	attreff_0dm3	オーディン3の周	fl_0dm3_010	fl_0dm3_010	fl_0dm3_010
16	attreff_beri	ベリ1の周	fl_beri_010	fl_beri_010	fl_beri_010
17	attreff_sgal	シガル1の周	fl_sgal_010	fl_sgal_010	fl_sgal_010
18	attreff_vpba	グレイブス1の周	fl_vpba_010	fl_vpba_010	fl_vpba_010
19	attreff_solsa	ソルサ1の周	fl_solsa_010	fl_solsa_010	fl_solsa_010
20	attreff_susei	水棲生物	fl_susei_010	fl_susei_010	fl_susei_010
21	attreff_sasa	ササ1の周	fl_sasa_010	fl_sasa_010	fl_sasa_010
22	attreff_suka	スケル1の周	fl_suka_010	fl_suka_010	fl_suka_010
23	attreff_0tyu	オチュー1の周	fl_0tyu_010	fl_0tyu_010	fl_0tyu_010
24	attreff_snsiba	サンシバ1の周	fl_snsiba_010	fl_snsiba_010	fl_snsiba_010
25	attreff_sokayu	ソクヤ1の周	fl_sokayu_010	fl_sokayu_010	fl_sokayu_010
26	attreff_sknmgf	スナミガキ1の周	fl_sknmgf_010	fl_sknmgf_010	fl_sknmgf_010
27	attreff_sknmgf	スナミガキ1の周	fl_sknmgf_010	fl_sknmgf_010	fl_sknmgf_010



# Let's have a listen.



# What? Wait a second!

- The methods for detailed calculations should be different for each type of character and creature.
  - Multi-legged creatures, birds, dragons, caterpillars...
- Are the numerical values minutely adjusted for each character?!

# What? Wait a second!

- The methods for detailed calculations should be different for each type of character and creature.
  - Multi-legged creatures, birds, dragons, caterpillars...
- Are the numerical values minutely adjusted for each character?!

Could the team even be using hard-coding?



# Specialized Script

- Designates the elements and equations to be used in determination processes
- Gives the parameters and threshold values

Script								
(	音源	センサー	種別	比較対象	条件	トリガ	前フレ	)
(	○	足 0	座標 (Y軸)	0.000000	接地	○		
	○	足 1	座標 (Y軸)	0.000000	接地	○		
	○	足 2	座標 (Y軸)	0.000000	接地	○		
	○	足 3	座標 (Y軸)	0.000000	接地	○		)
		移動速度	速度 (Z軸)	10.000000	小さい <			
(		移動方向と向きの差分	座標 (Y軸)	-1.570000	大きい >			
		移動方向と向きの差分	座標 (Y軸)	1.570000	小さい <			)
								結合条件
								OR(もしくは)
								OR(もしくは)
								OR(もしくは)
								AND(かつ)
								AND(かつ)
								AND(かつ)
								OR(もしくは)

# Specialized Script

- Designates the elements and equations to be used in determination processes
- Gives the parameters and threshold values
- **Complete tool for sound designers!**
- **Data-driven**

# Script Tool

ffxiii  
AutoSE Editor

<< >> Play

種別選択

種別名  
標準2足  
標準4足  
標準羽型  
1本足  
6本足

Script 効果音種別

部位名	サイズ
歩き	68 byte
走り	68 byte
ジャンプ	48 byte
落地	36 byte
スライディング	56 byte
後退	56 byte
しゃがみ	40 byte
立ち上がり	40 byte
腕振り	24 byte
拳手	24 byte
羽音1	0 byte
羽音2	0 byte

MAPSET <mset\_hang\_0  
CHRSET <chset\_2016\_0

Script

音源	センサー	種別	比較対象	条件	トリガ	前フレ	結合条件
足 0	足 0	座標 (Y軸)	0.000000	接地	○		OR(もしくは)
移動速度	移動速度	座標 (Z軸)	10.000000	小さい <			AND(かつ)
移動方向と向きの差分	移動方向と向きの差分	座標 (Y軸)	-1.570000	大きい >			AND(かつ)
移動方向と向きの差分	移動方向と向きの差分	座標 (Y軸)	1.570000	小さい <			OR(もしくは)

条件変更

センサー 種別

移動方向と向きの差分 座標(Y軸)

比較対象

数値 -1.570000

条件

大きい >

追加 キャンセル

CHR

CHR

Info

■現在のフレーム  
進行方向 -2.748889  
向きと進行方向の差分 0.000000  
移動速度 0.000000  
経過時間 0.050050

本体::POS 254.948371, 4.090579, 587.979816  
本体::ROT -0.000000, -2.748889, -0.000000  
本体::SPEED 0.000000, 0.000000, 0.000000, 0.000000  
本体::ACC 0.000000, 0.000000, 0.000000, 0.000000  
足(0)::POS -0.126088, -1.067932, -0.115707  
足(0)::ROT 1.336204, 1.656388, 1.591444  
足(0)::SPEED 0.006402, 0.003138, 0.006851, 0.008888  
足(0)::ACC 0.004682, -0.000089, -0.003361, -0.000952  
踵(0)::POS -0.138689, -0.553129, 0.007327  
踵(0)::ROT 2.820335, -1.051528, -2.935196  
踵(0)::SPEED 0.002346, 0.002367, 0.008532, 0.009159  
踵(0)::ACC -0.001014, -0.001320, 0.000420, -0.000364  
脛(0)::POS -0.058071, -0.140030, -0.028564  
脛(0)::ROT -1.484131, 2.709516, -1.756546  
脛(0)::SPEED 0.011472, 0.003704, 0.004752, 0.012958  
脛(0)::ACC -0.004722, -0.001783, -0.002796, -0.005792  
足(1)::POS 0.119705, -1.071282, -0.094171  
足(1)::ROT 1.398055, 1.900763, 1.630489  
足(1)::SPEED 0.005736, 0.003138, 0.002376, 0.006957  
足(1)::ACC -0.008430, 0.002315, -0.006012, -0.003508  
踵(1)::POS 0.088935, -0.551355, -0.003453  
踵(1)::ROT 3.065131, 3.080945, -3.121654  
踵(1)::SPEED 0.008673, 0.010872, 0.057768, 0.059572  
踵(1)::ACC -0.002028, -0.002778, 0.000840, -0.000127  
脛(1)::POS 0.071943, -0.131774, 0.037070  
脛(1)::ROT -1.687076, -1.180182, -1.531049  
脛(1)::SPEED 0.004404, 0.004493, 0.006576, 0.009101  
脛(1)::ACC -0.001362, -0.002281, -0.005316, -0.005750  
肘(0)::POS -0.112434, 0.044197, -0.162977  
肘(0)::ROT 1.522773, 0.236710, 1.374145  
肘(0)::SPEED 0.006897, 0.004459, 0.009792, 0.010782  
肘(0)::ACC -0.007415, -0.002195, 0.006432, -0.000234  
肩(0)::POS -0.060495, 0.304901, -0.150448  
肩(0)::ROT -1.618819, -2.904882, -1.767448  
肩(0)::SPEED 0.008748, 0.004047, 0.015385, 0.018155



# Script Details

- Elements
  - Bone designation, input values (16 types)
  - Coordinates, angle, speed, acceleration (4 types)
  - XYZ (3 types)
- Comparative code
  - 14 types (==, >=, ||, etc.)
- Unique specifications
  - Order of calculation priorities, comparisons with previous states and values, etc.

# Examples of Script Use #1

- Swinging of Arms

( Angle from side of right shoulder > -1.4 & previously was not like this ) | |

( Angle from side of left shoulder > -1.4 & previously was not like this )

Script								
(	音源	センサー	種別	比較対象	条件	トリガ	前フレ	) 結合条件
	○	肩 0	角度 (ZY面:横が	-1.400000	大きい >	○		OR(もしくは)
	○	肩 1	角度 (ZY面:横が	-1.400000	大きい >	○		OR(もしくは)

# Examples of Script Use #2

- Raising of Arms

( Angle from side of right shoulder > 0.4 & previously was not like this ) | |

( Angle from side of left shoulder > 0.4 & previously was not like this )

Script								
(	音源	センサー	種別	比較対象	条件	トリガ	前フレ )	結合条件
	○	肩 1	角度 (ZY面:横か	0.400000	大きい >	○		OR(もしくは)
	○	肩 0	角度 (ZY面:横か	0.400000	大きい >	○		OR(もしくは)



# Examples of Script Use #3

- Sliding

( ( Right leg == in contact with terrain &  
Left leg == in contact with terrain ) &  
previously was also like this ) &  
( Character movement speed  $\geq 10$  ) &  
( ( Character acceleration  $< -2.5$  ) &  
previously was not like this )

Script								
(	音源	センサー	種別	比較対象	条件	トリガ	前フレ	)
(		足 1	速度 (Y軸)	0.000000	接地		○	AND(かつ)
		足 0	速度 (Y軸)	0.000000	接地		○	) AND(かつ)
(		移動速度	速度 (Y軸)	10.000000	以上 $\geq$			AND(かつ)
		本体	加速度 (3D)	-2.500000	以下 $\leq$	○		) OR(もしくは)

# Examples of Script Use #4

- Crouching

( Right leg == in contact with terrain &  
left leg == in contact with terrain ) &  
( ( Right knee Y axis position > Right hip ) |  
( Left knee Y axis position > Left hip ) ) &  
previously was not like this )

Script									
(	音源	センサー	種別	比較対象	条件	トリガ	前フレ	)	結合条件
(		足 0	座標 (Y軸)	0.000000	接地				AND(かつ)
		足 1	座標 (Y軸)	0.000000	接地			)	AND(かつ)
		膝 0	座標 (Y軸)	腰 0	大きい >	○			OR(もしくは)
		膝 1	座標 (Y軸)	腰 1	大きい >	○			OR(もしくは)

# Problems and Solutions

- Everything A-OK, right? ... Wrong!
  - Problems with cutscenes
  - Problems with physics results
  - Unforeseen character actions and numerical values
  - Problems finding the area that needed to be adjusted





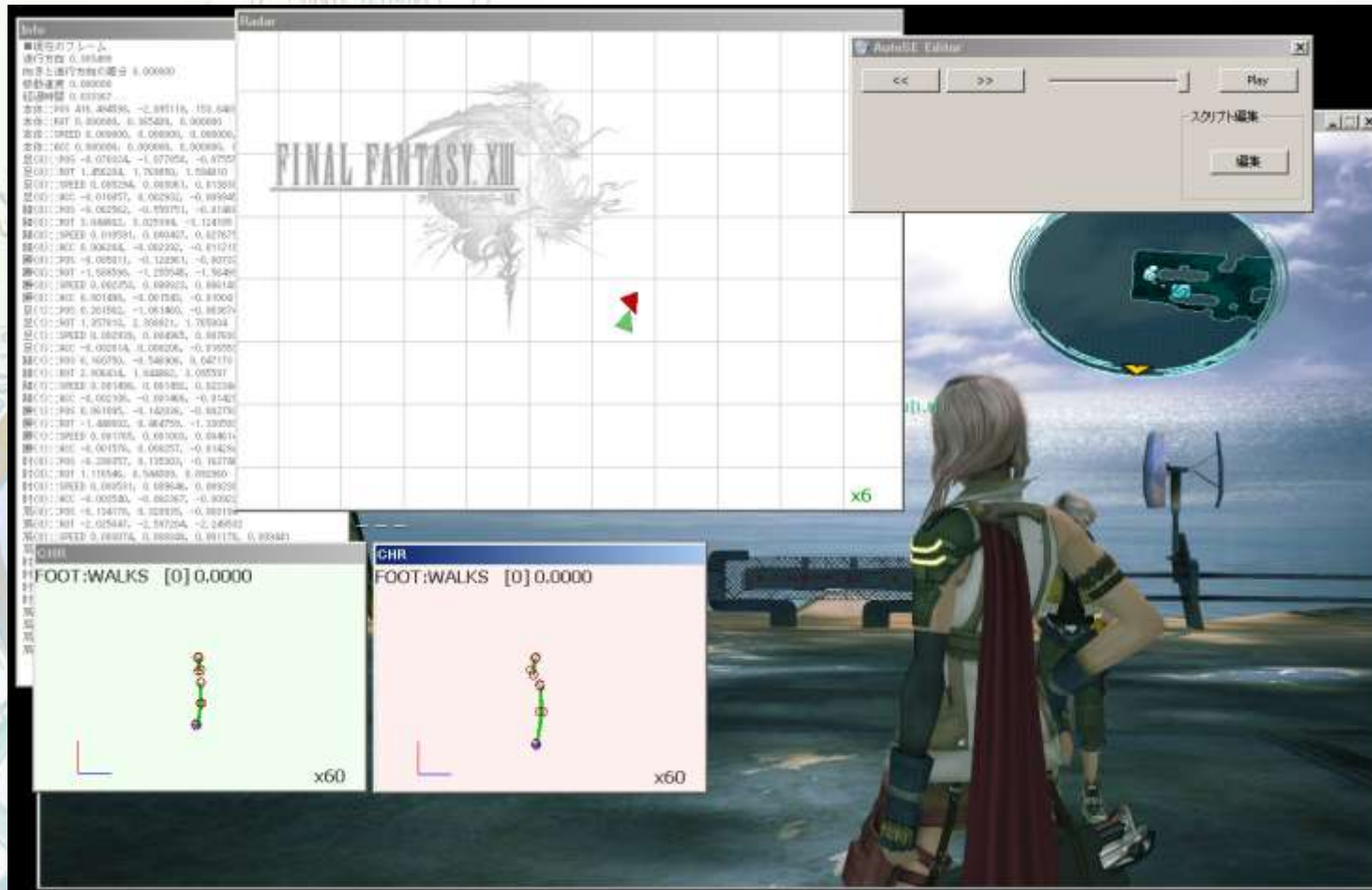
# Troublesome characters



# Solutions

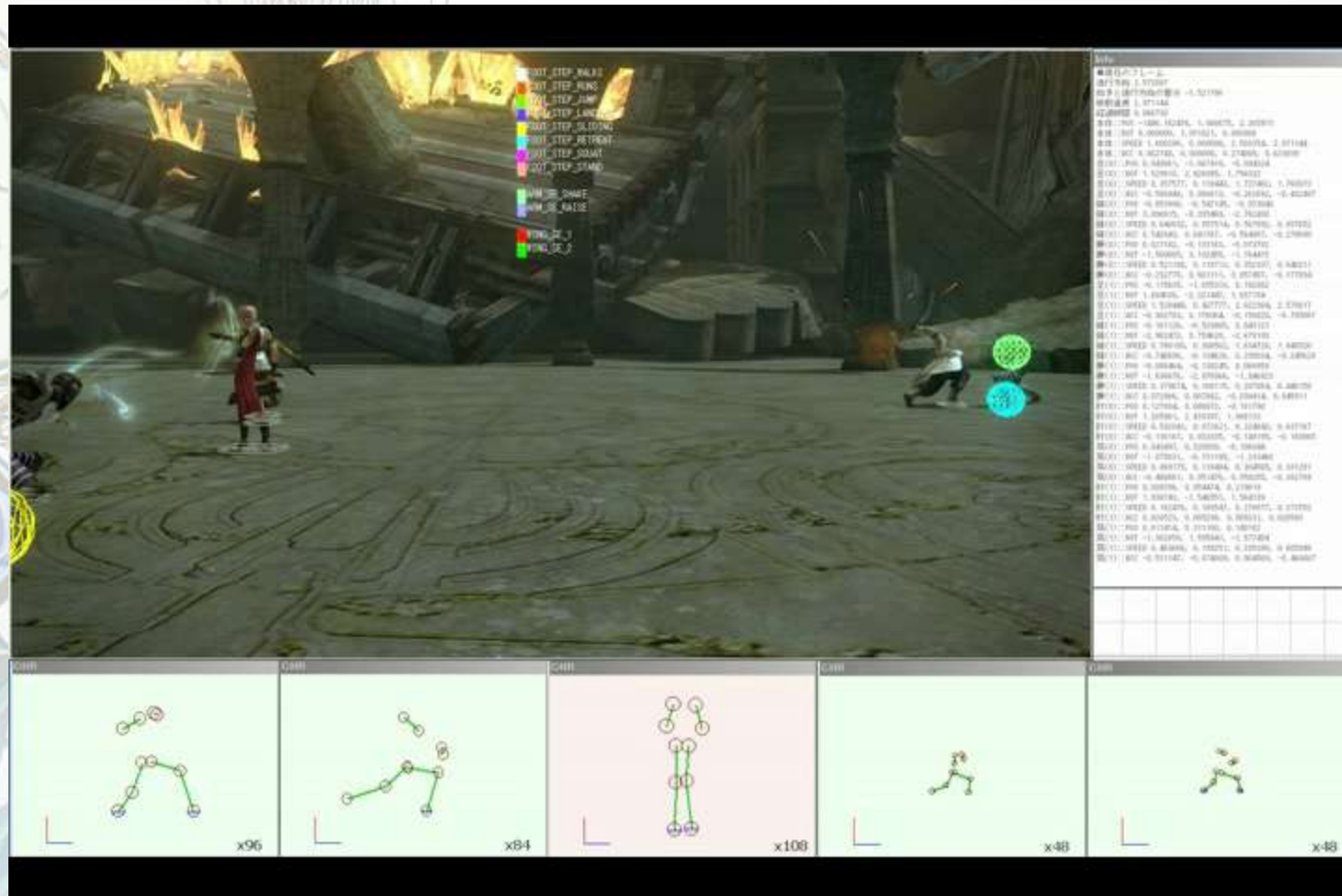
- Created a method for automatically detecting irregularities
- Made it possible to use the traditional way of designating sound effects by hand alongside automation
- Created a pre-rendering mechanism

# Specialized Debugger





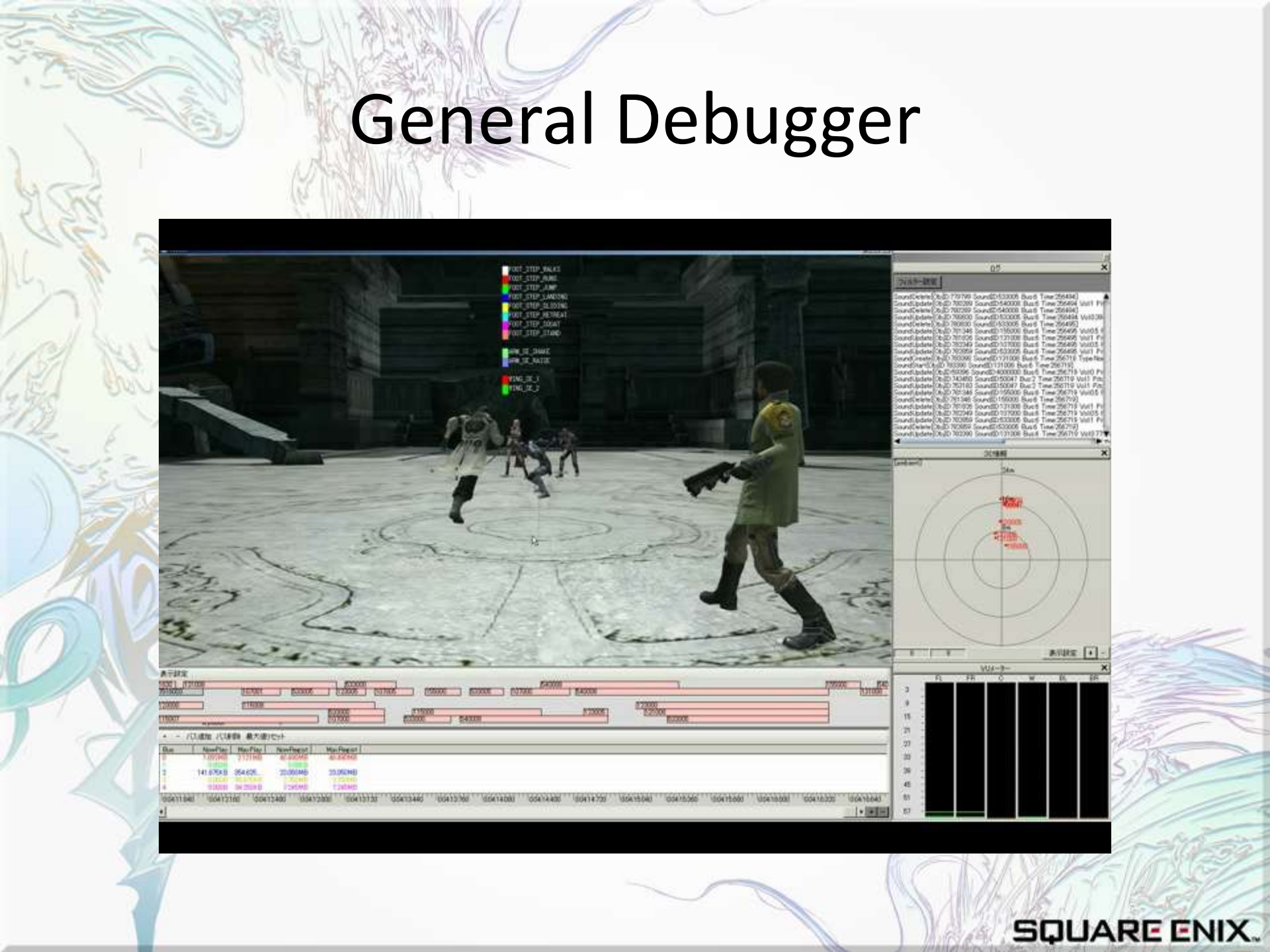
# Specialized Debugger



# General Debugger

The screenshot displays the General Debugger interface for a game. The main window shows a 3D game scene with a character in a green uniform and several enemies in a dark, industrial environment. On the left, a list of active sounds is shown with color-coded bars. On the right, a radar displays the positions of enemies. At the bottom, a performance table shows various metrics for the game engine.

Item	Min	Max	Avg	Max
00000000	0.000000	0.000000	0.000000	0.000000
00000001	0.000000	0.000000	0.000000	0.000000
00000002	0.000000	0.000000	0.000000	0.000000
00000003	0.000000	0.000000	0.000000	0.000000
00000004	0.000000	0.000000	0.000000	0.000000
00000005	0.000000	0.000000	0.000000	0.000000
00000006	0.000000	0.000000	0.000000	0.000000
00000007	0.000000	0.000000	0.000000	0.000000
00000008	0.000000	0.000000	0.000000	0.000000
00000009	0.000000	0.000000	0.000000	0.000000
0000000A	0.000000	0.000000	0.000000	0.000000
0000000B	0.000000	0.000000	0.000000	0.000000
0000000C	0.000000	0.000000	0.000000	0.000000
0000000D	0.000000	0.000000	0.000000	0.000000
0000000E	0.000000	0.000000	0.000000	0.000000
0000000F	0.000000	0.000000	0.000000	0.000000
00000010	0.000000	0.000000	0.000000	0.000000
00000011	0.000000	0.000000	0.000000	0.000000
00000012	0.000000	0.000000	0.000000	0.000000
00000013	0.000000	0.000000	0.000000	0.000000
00000014	0.000000	0.000000	0.000000	0.000000
00000015	0.000000	0.000000	0.000000	0.000000
00000016	0.000000	0.000000	0.000000	0.000000
00000017	0.000000	0.000000	0.000000	0.000000
00000018	0.000000	0.000000	0.000000	0.000000
00000019	0.000000	0.000000	0.000000	0.000000
0000001A	0.000000	0.000000	0.000000	0.000000
0000001B	0.000000	0.000000	0.000000	0.000000
0000001C	0.000000	0.000000	0.000000	0.000000
0000001D	0.000000	0.000000	0.000000	0.000000
0000001E	0.000000	0.000000	0.000000	0.000000
0000001F	0.000000	0.000000	0.000000	0.000000
00000020	0.000000	0.000000	0.000000	0.000000
00000021	0.000000	0.000000	0.000000	0.000000
00000022	0.000000	0.000000	0.000000	0.000000
00000023	0.000000	0.000000	0.000000	0.000000
00000024	0.000000	0.000000	0.000000	0.000000
00000025	0.000000	0.000000	0.000000	0.000000
00000026	0.000000	0.000000	0.000000	0.000000
00000027	0.000000	0.000000	0.000000	0.000000
00000028	0.000000	0.000000	0.000000	0.000000
00000029	0.000000	0.000000	0.000000	0.000000
0000002A	0.000000	0.000000	0.000000	0.000000
0000002B	0.000000	0.000000	0.000000	0.000000
0000002C	0.000000	0.000000	0.000000	0.000000
0000002D	0.000000	0.000000	0.000000	0.000000
0000002E	0.000000	0.000000	0.000000	0.000000
0000002F	0.000000	0.000000	0.000000	0.000000
00000030	0.000000	0.000000	0.000000	0.000000
00000031	0.000000	0.000000	0.000000	0.000000
00000032	0.000000	0.000000	0.000000	





# Results

- Work required for manual assignment of sounds was reduced and work required for cutscenes was reduced to one third
- The motion team and planners can freely edit the game without worrying about the sound department!
- Capable of creating minute sound effects based on complicated user actions
- Capable of reacting smoothly to motion changes in real time



# Current Projects

- Enrichment of automation patterns in various applications (such as gear and vehicles)
- Auto-adjustments to the appropriation of data templates and bone IDs for new models
- Developing a compact engine that would take the place of table data

# Q&A



©SQUARE ENIX CO., LTD. All Rights Reserved.

**SQUARE ENIX**