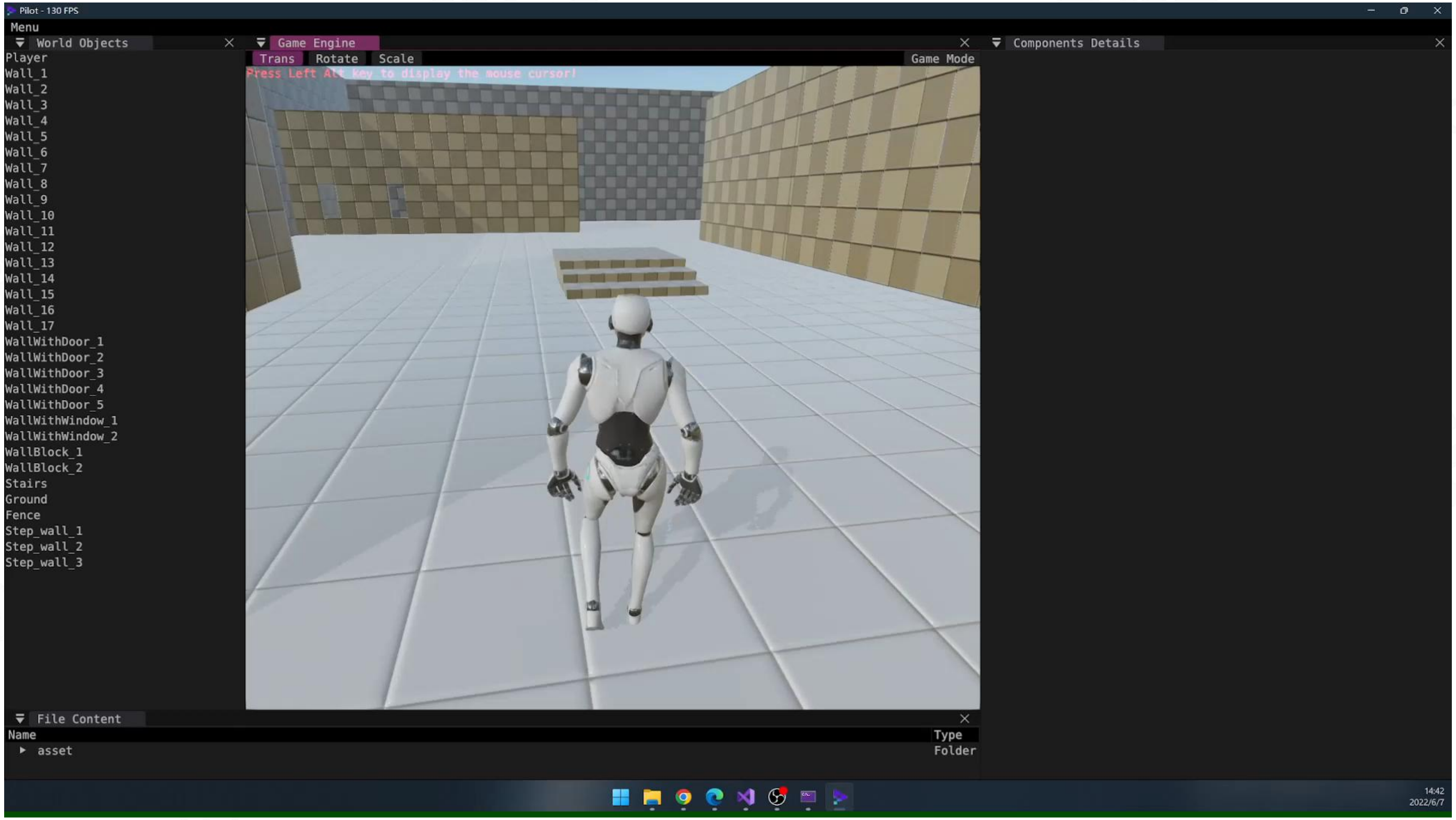
Voice from Communities

- Piccolo Game Engine Logo
- Piccolo is from Italian, meaning “short flute”. Although it’s small, Piccolo is the instrument with the highest pitch, representing that despite our engine is small, with everyone’s joint efforts, it could bring the biggest energy
- The hole of the flute looks like “0” and the body of flute, looks like “1”, representing the computing world
- We are replacing Pilot with Piccolo in GitHub files, Piccolo website and BBS coming soon
- Thank you for staying with us, good luck with finals, graduation, work!

Homework 3 Submissions

- Homework 3 due time, July 14th 00:00

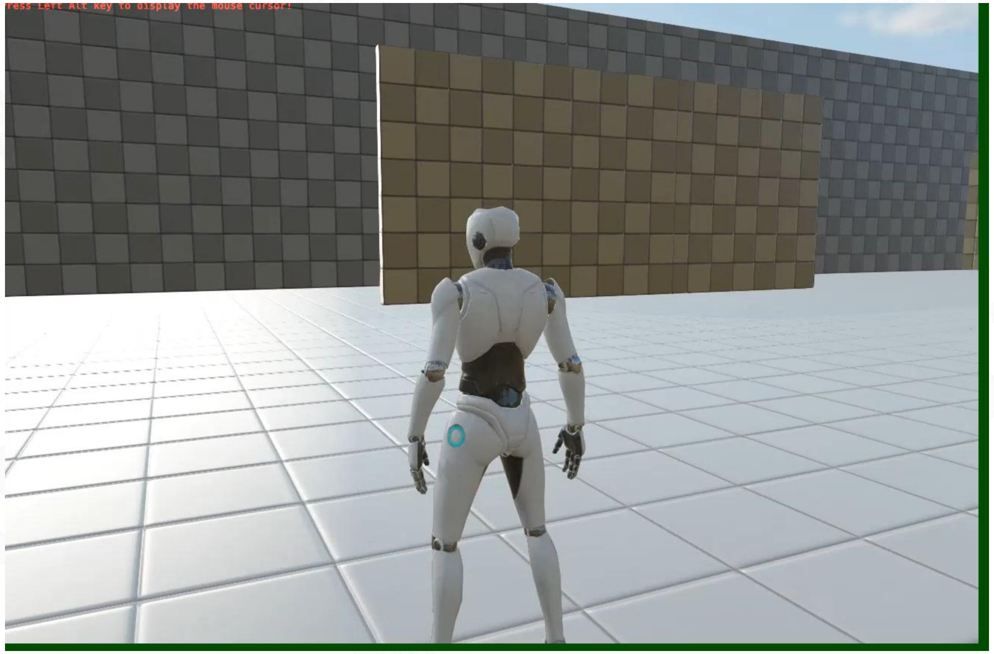


@翁同学

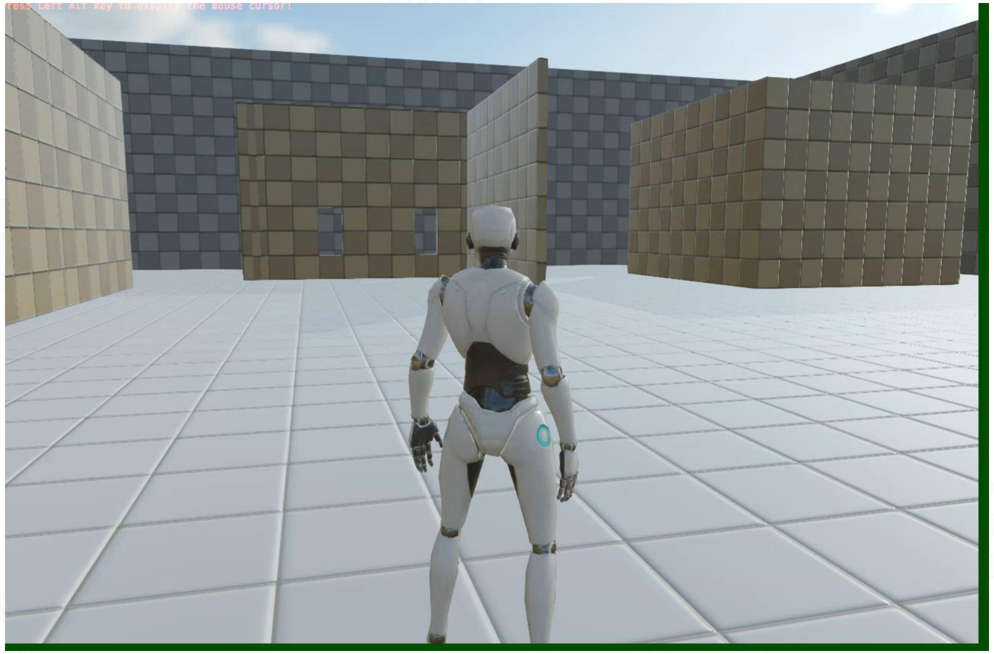
Notes on Animation Homework

- Blend
  - Weights need to be normalized
  - The shortest path flag of NLerp

```
static Quaternion nLerp(float t, const Quaternion& kp, const Quaternion& kq, bool shortest_path = false);
```



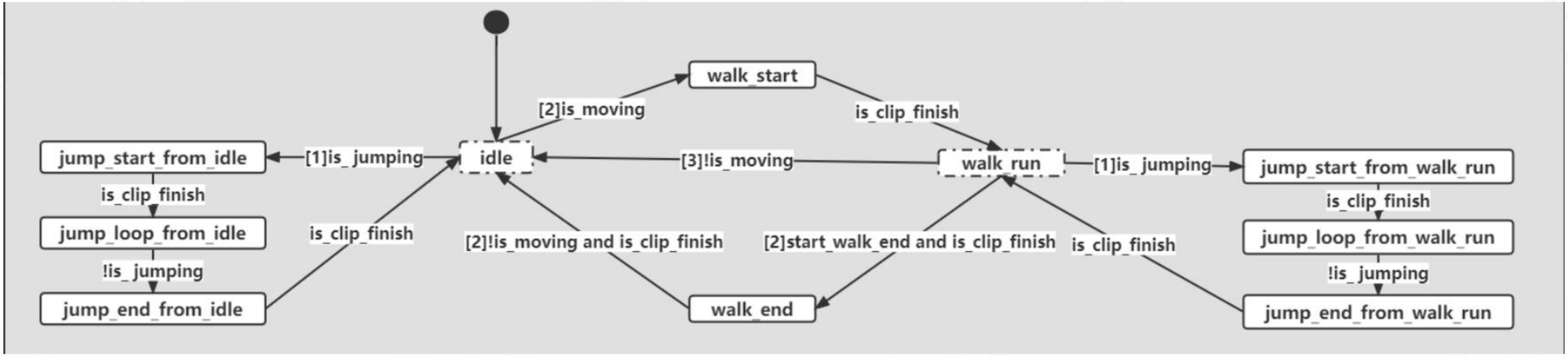
Weights not be normalized



The Shortest path flag error

Notes on Animation Homework

- ASM
  - Focus on the implementation of ASM logic
  - May have sliding step problem due to limited animation resource



Lecture 12

Effects

Modern Game Engine - Theory and Practice

WANG XI

GAMES 104

2022


93694
Smokin' Sexy Style!!
Devil May Cry 5


Elden Ring


Final Fantasy XVI


Sekiro: Shadows Die Twice

History of Particle System

“A particle system is a collection of many many minute particles that together represent a fuzzy object. Over a period of time, particles are generated into a system, move and change from within the system, and die from the system.”

—William Reeves, "Particle Systems—A Technique for Modeling a Class of Fuzzy Objects," ACM Transactions on Graphics 2:2 (April 1983), 92.

Star Trek II: The Wrath of Khan
(first introduced particle system, 1982)

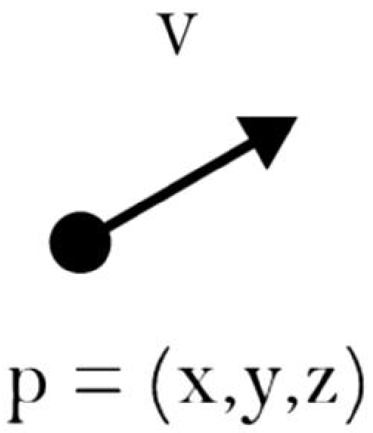
Particle

A particle in game is usually a sprite or 3D model,

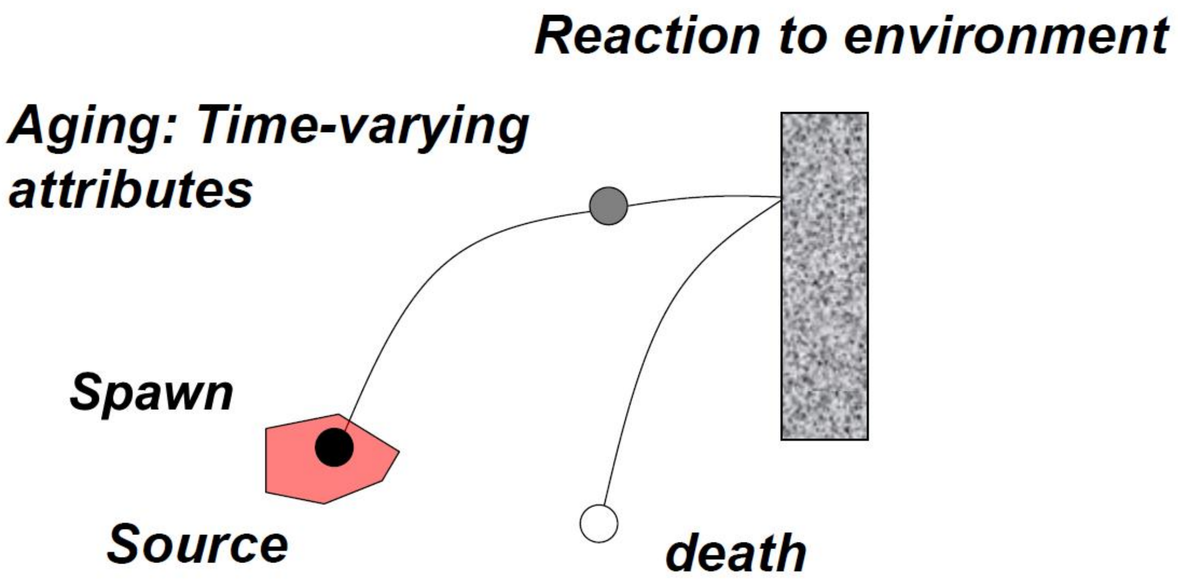
with the following attributes:

- Position
- Velocity
- Size
- Color
- Lifetime

……



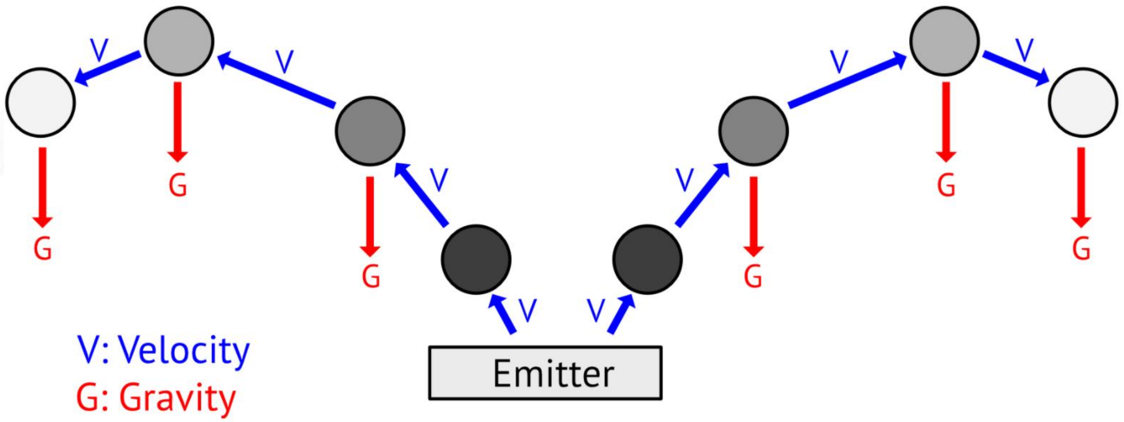
Particle's Life Cycle



Particle Emitter

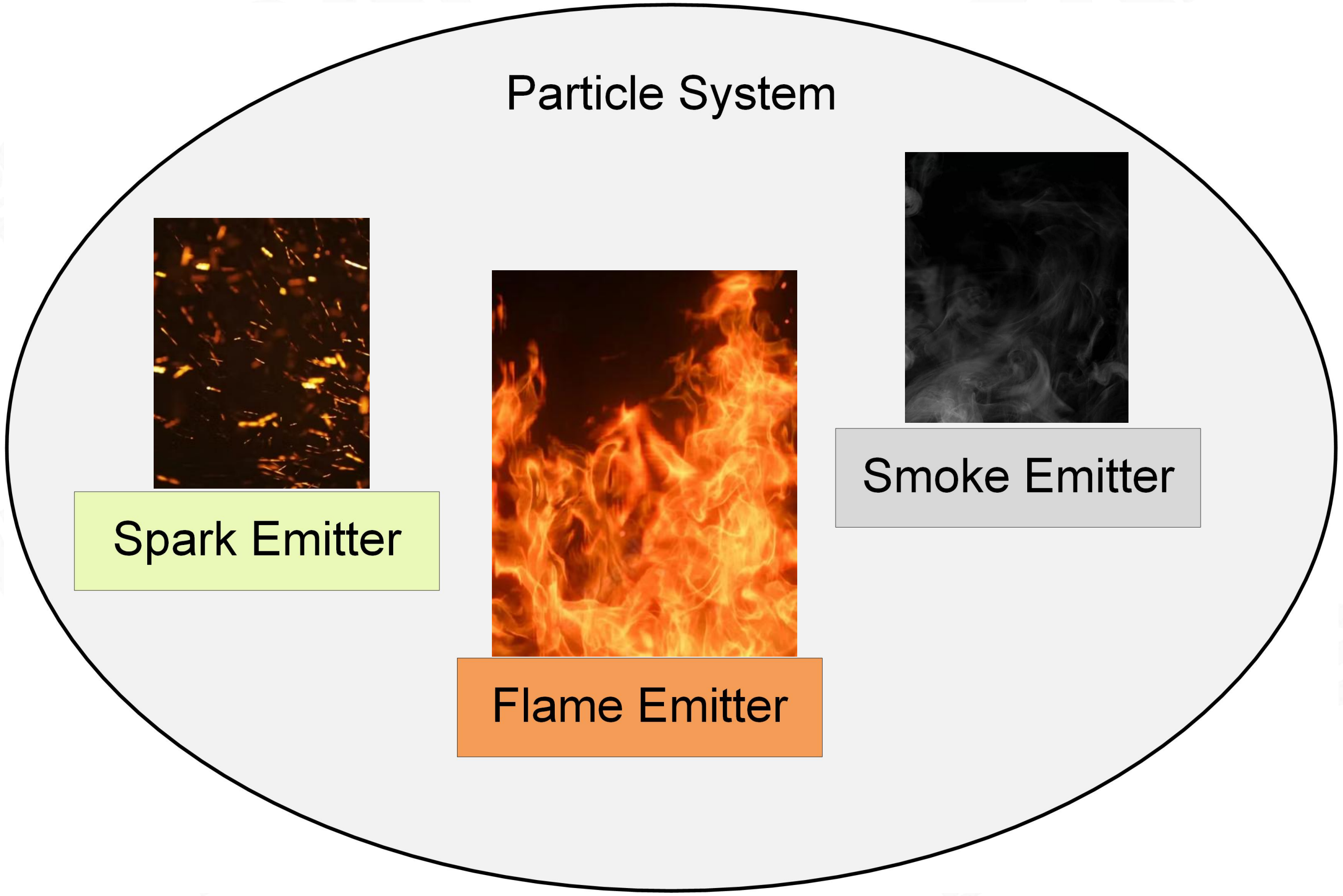
Particle Emitter is used to define the particles simulation

- Specify the spawn rules
- Specify simulation logic
- Describe how to render particles



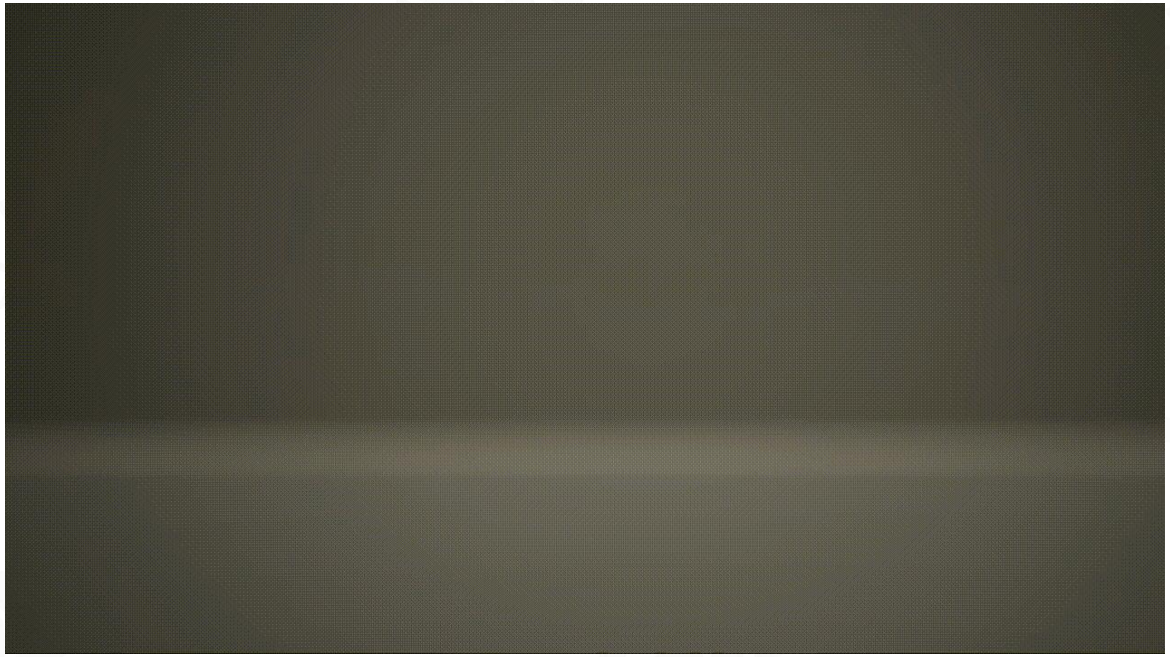
Particle System

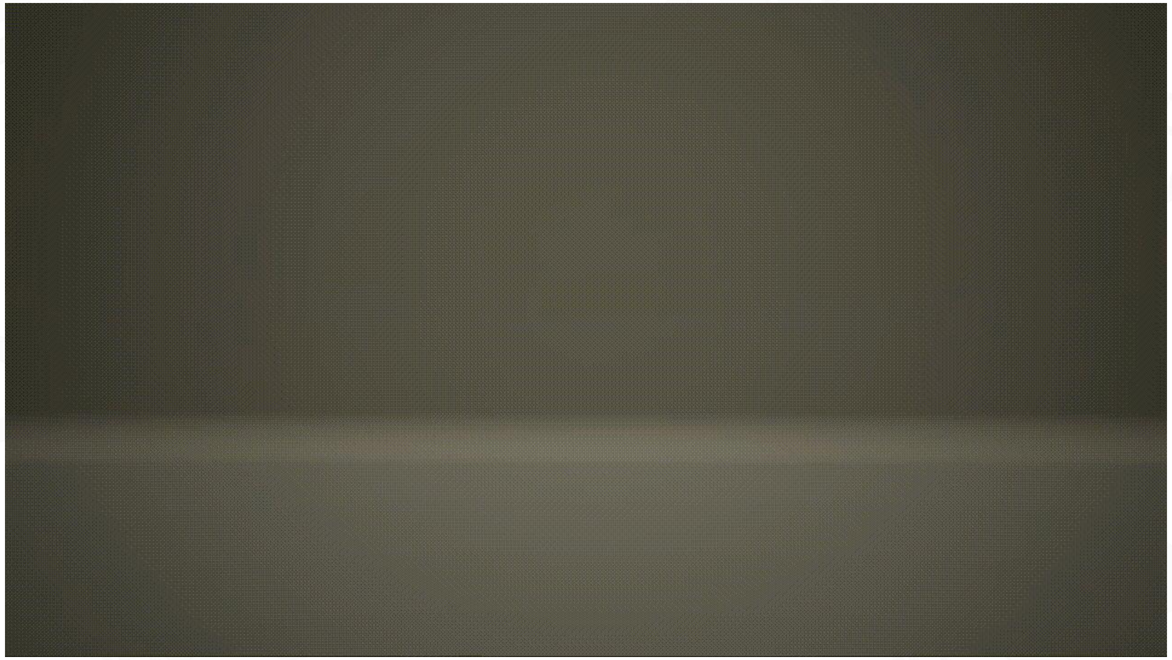
A particle system is a collection of individual emitters



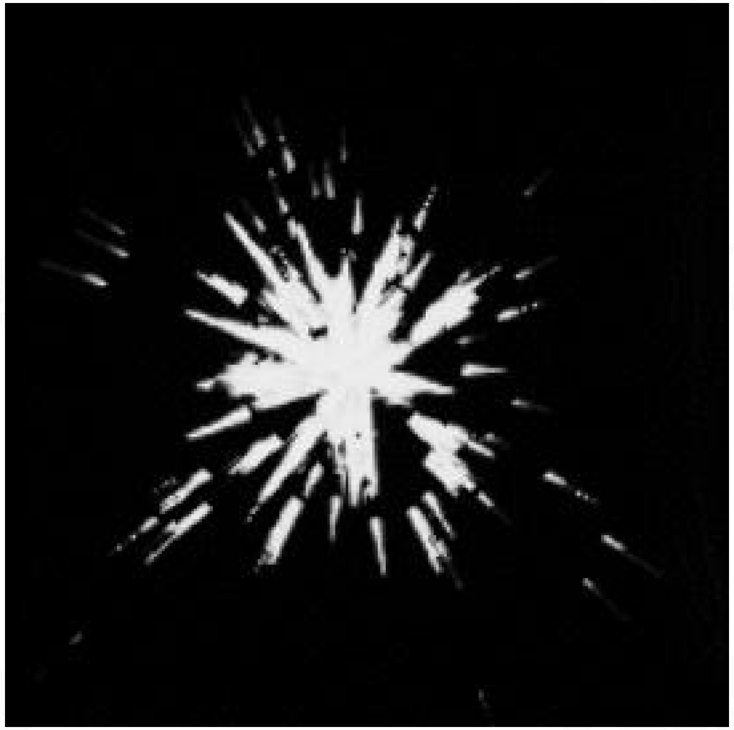
Particle System

A simple case, just combine different emitters

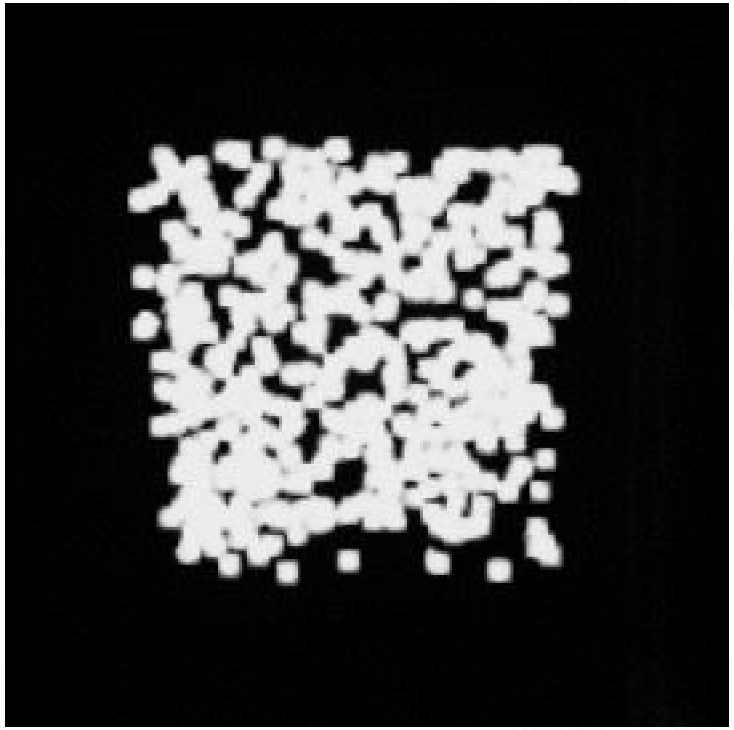




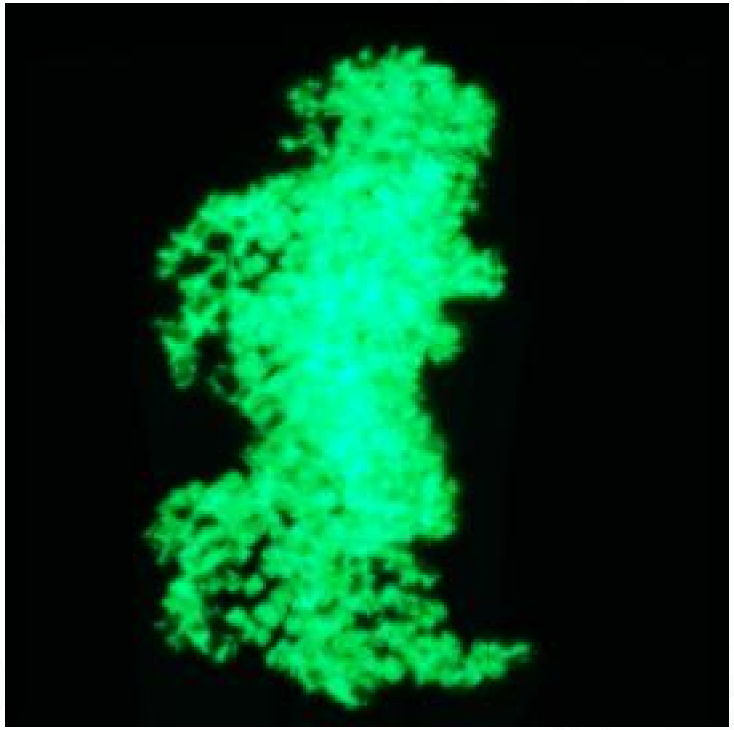
Particle Spawn Position



Single position spawn



Spawn based on area



Spawn based on mesh

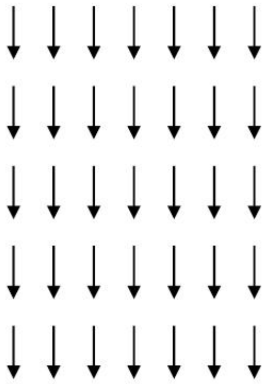
Particle Spawn Mode

Continuous
- variable spawn rate per frame
- time, distance based, etc.

Burst
- all particle spawn and simulated at once.

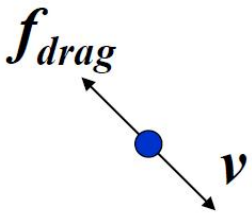
Simulate (1/5)

Common forces



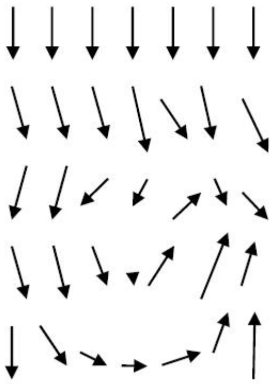
Gravity

$f = mg$



Viscous Drag

$f = -k_d v$

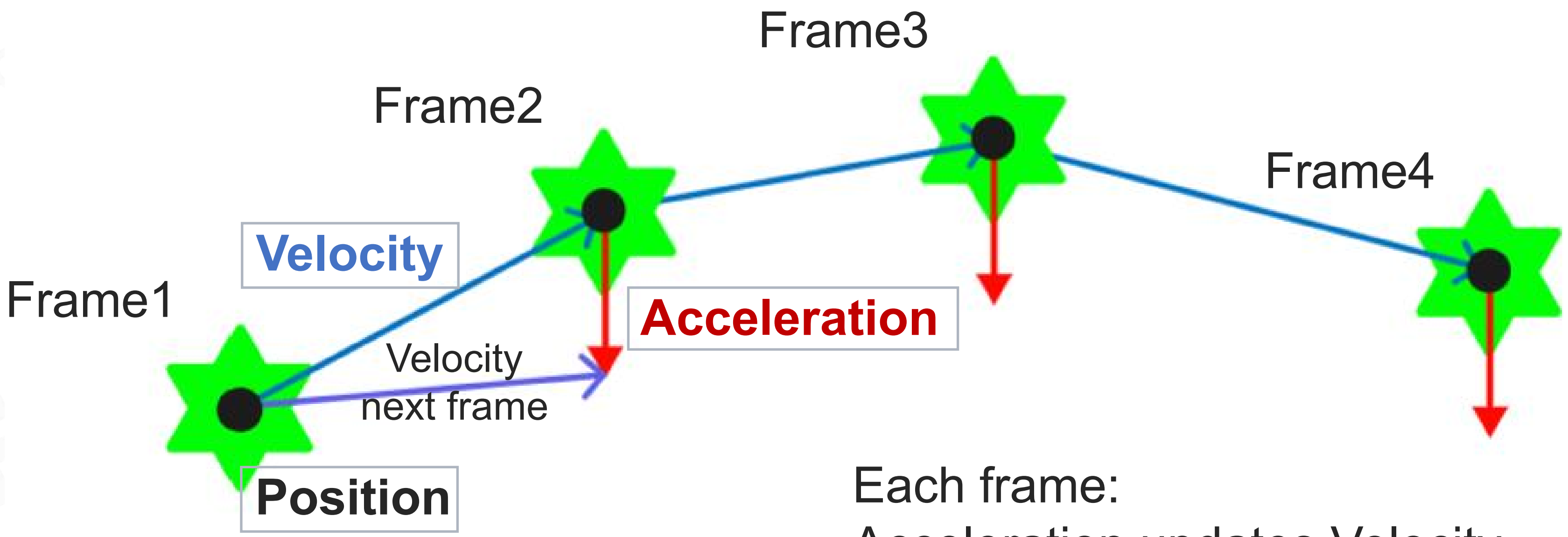


Wind Fields

$f = kv_{wind}$

Simulate (2/5)

Simulate controls how the particles change over time



Each frame:
Acceleration updates Velocity
Velocity updates postion

Simulate (3/5)

Simulate gravity

Simulate gravity & rotation

Simulate (4/5)

Simulate gravity & color

Simulate gravity & size

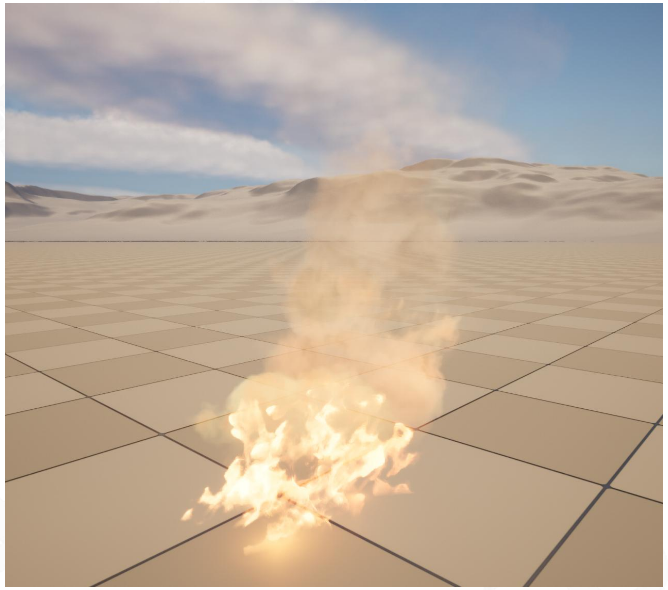
Simulate (5/5)

Without collision

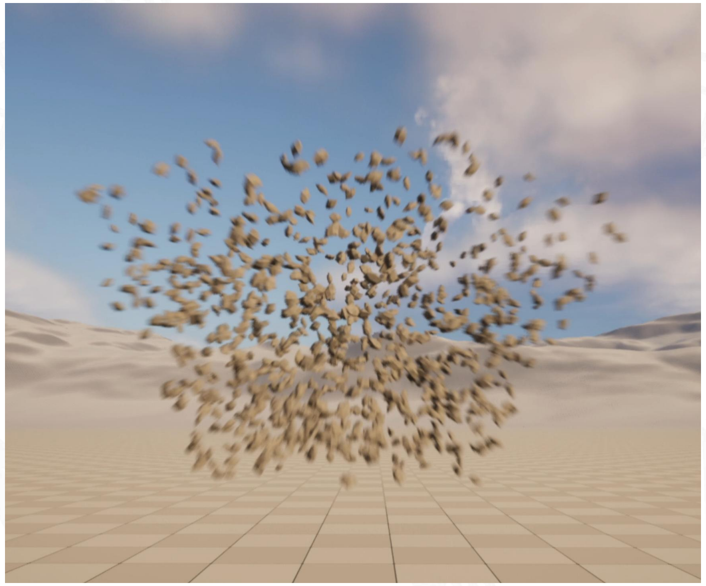
With collision

Particle Type

- Billboard Particle
- Mesh Particle
- Ribbon Particle



Billboard Particle



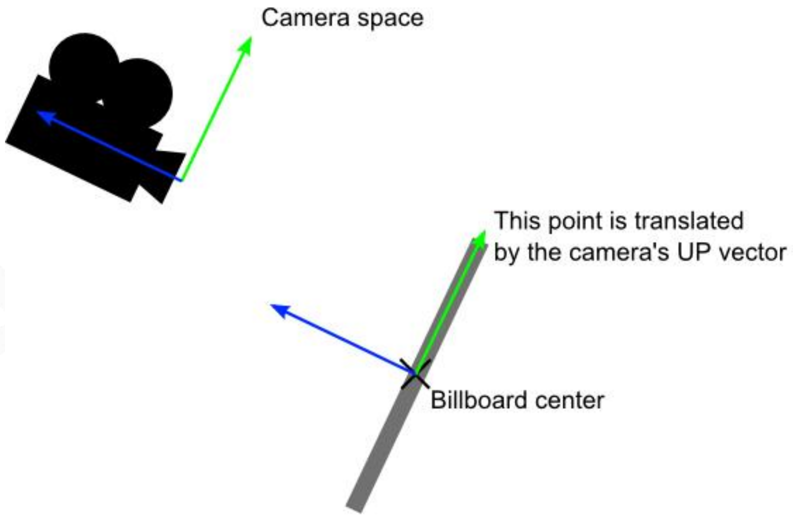
Mesh Particle

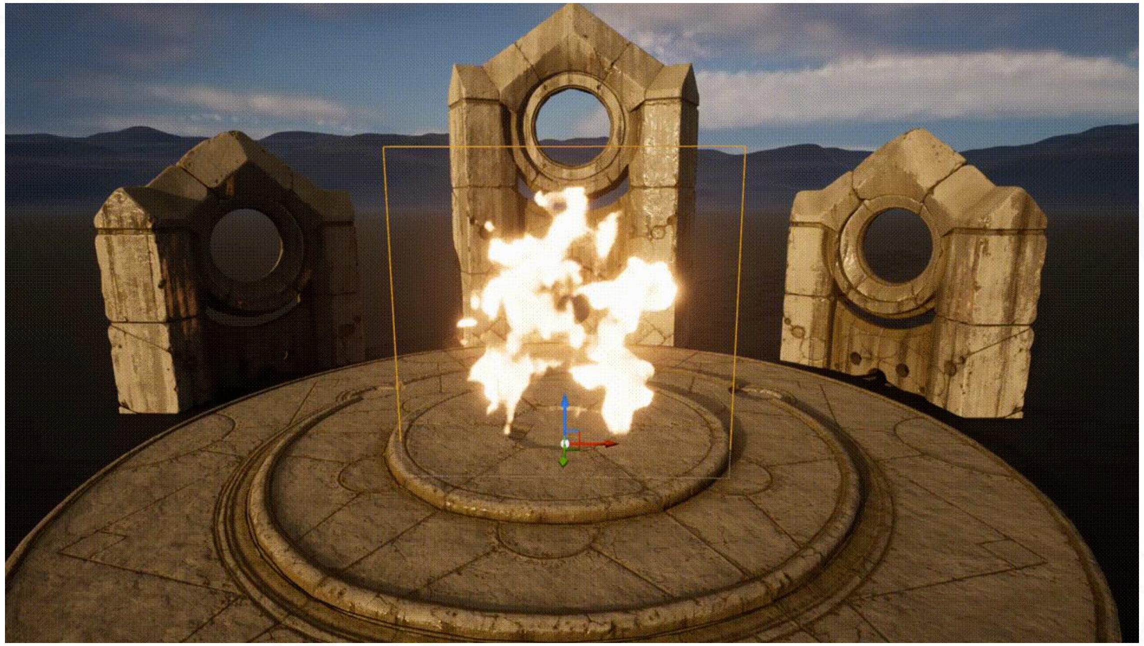
Ribbon Particle

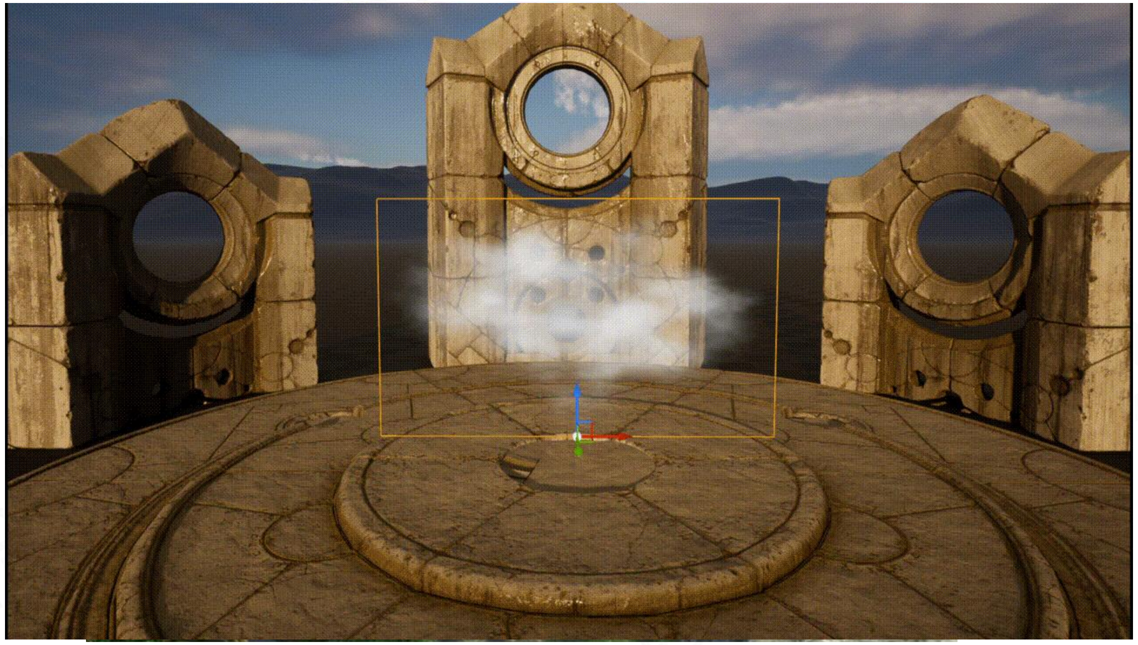
Billboard Particle

Each particle is a sprite

- Appears to be 3D
- Always face the camera



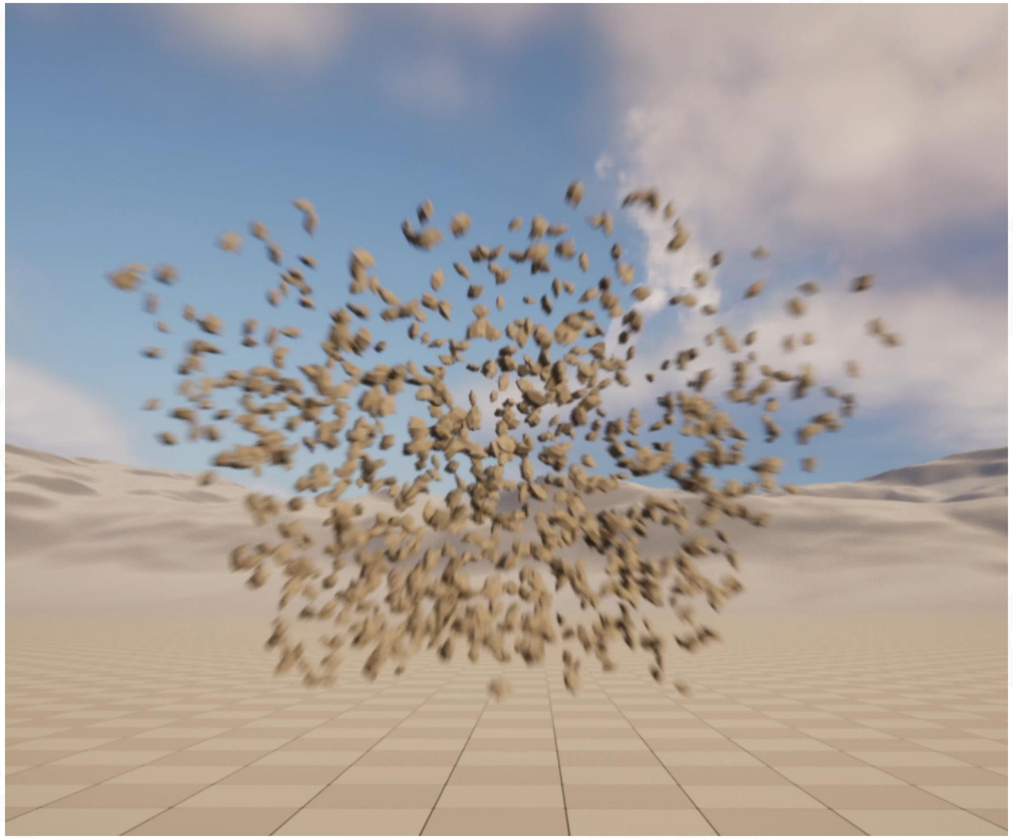




Mesh Particle

Each particle is a 3D model.

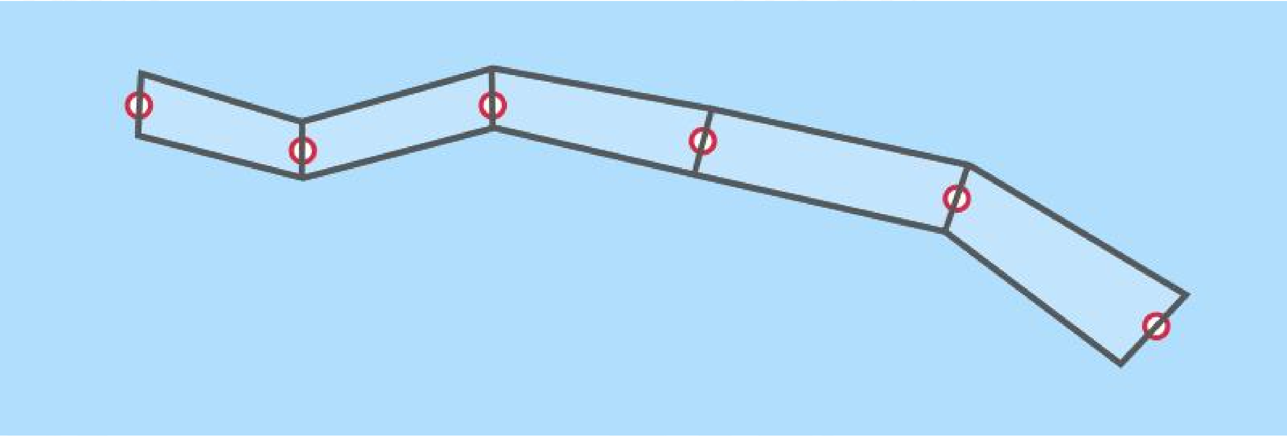
- Rocks coming from a explosion



Ribbon Particle

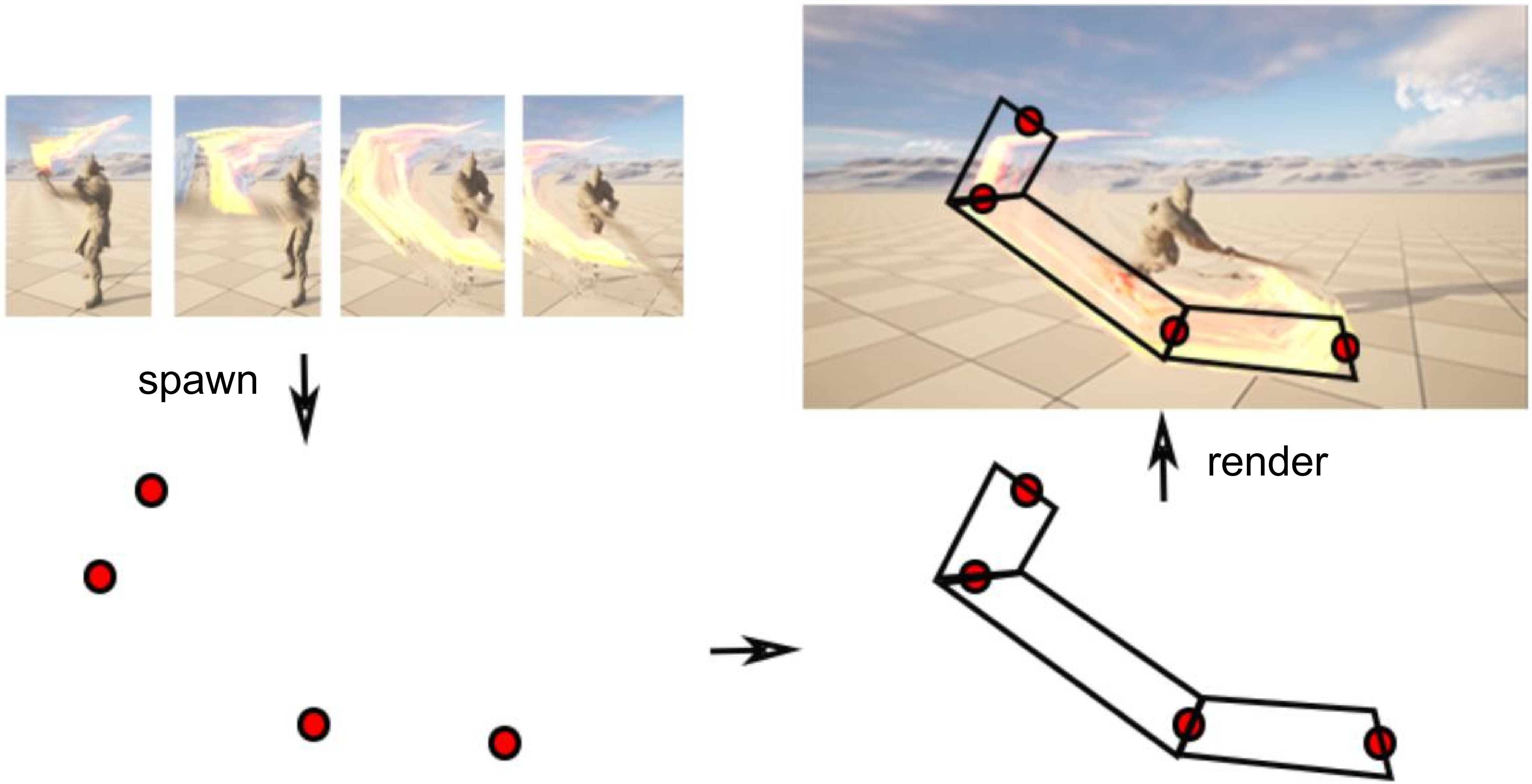
A strip is created by connecting the particles and rendering quads between the adjacent particles.

- particles (represented as red dots)



Ribbon Particle:Slash

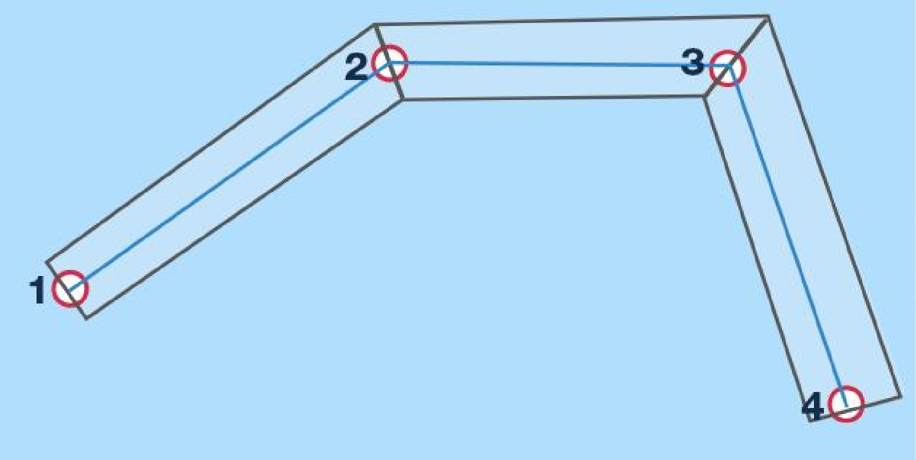
Ribbon Particle Case(1/3)



Ribbon Particle Case(2/3)

No smoothing shape

- with sharp angles

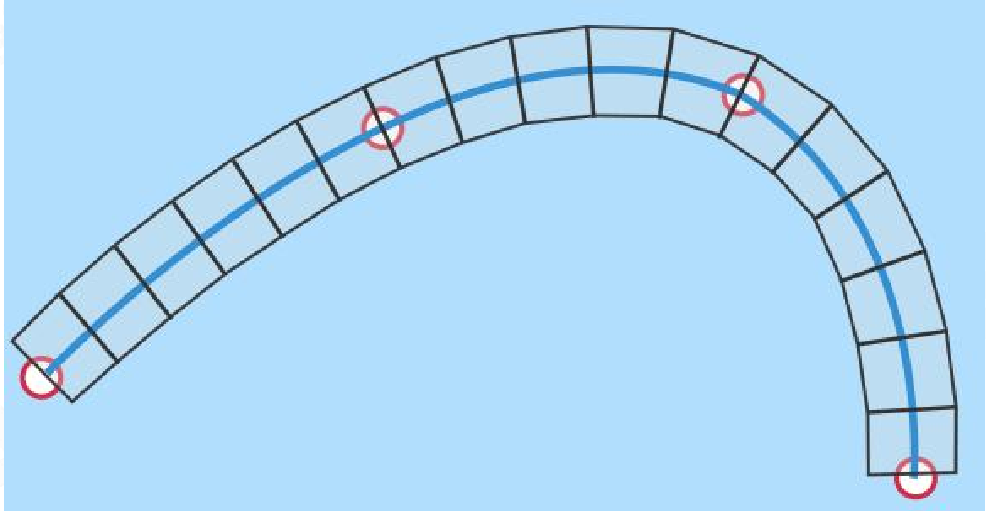


Slash without smoothing

Ribbon Particle Case(3/3)

Smoothing with Centripetal Catmull-Rom interpolation

- add extra segments between particles
- can set the number of segments
- requires more CPU



smoothing with Centripetal Catmull-Rom

Particle System Rendering

Alpha Blending Order

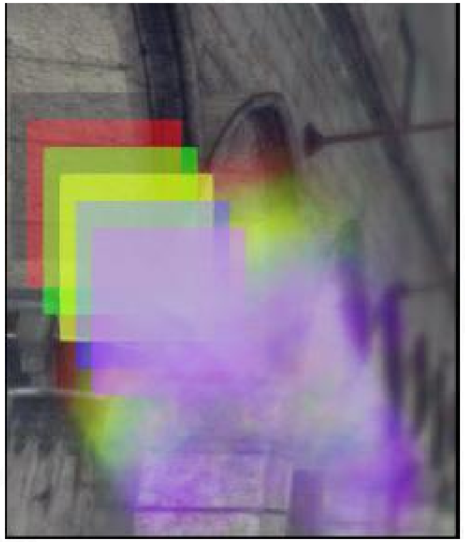
$$C_f = a_1 C_1 + (1 - a_1) C_0$$

Final color

Source color

Destination color

Blend result of unsorted elements



Blend result of sorted elements

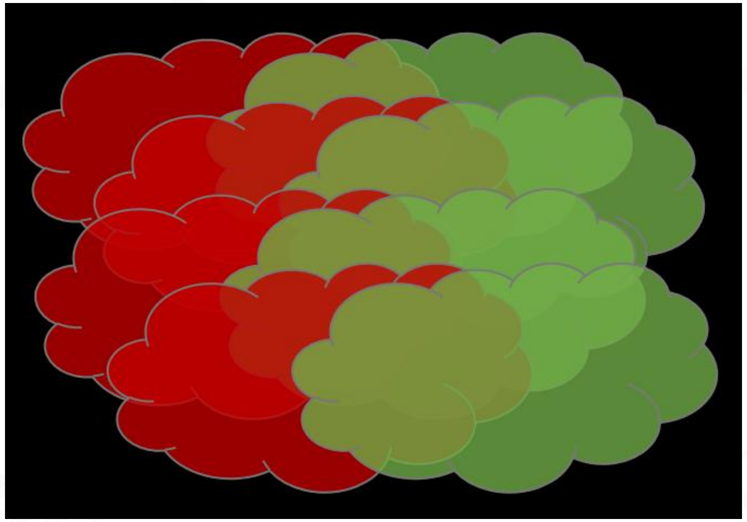
Particle Sort

Sorting mode

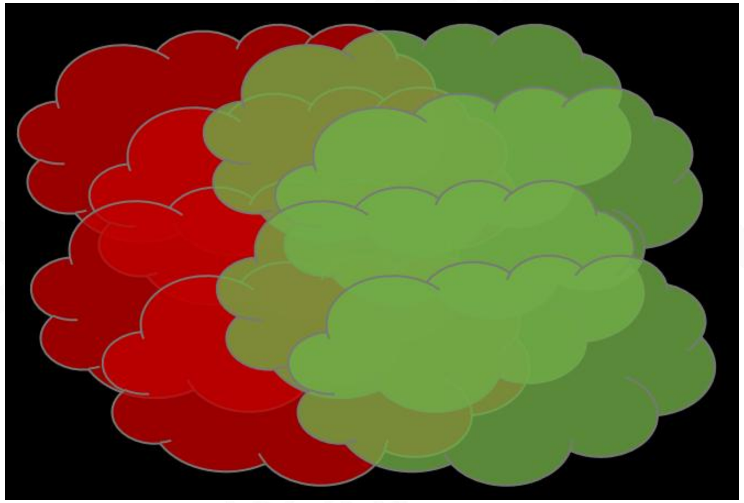
- Global
  - Accurate, but large performance consumption
- Hierarchy: Per system -> Per emitter -> Within emitter

Sort rules

- Between particles: based on particle distance with camera
- Between systems or emitters: bounding box

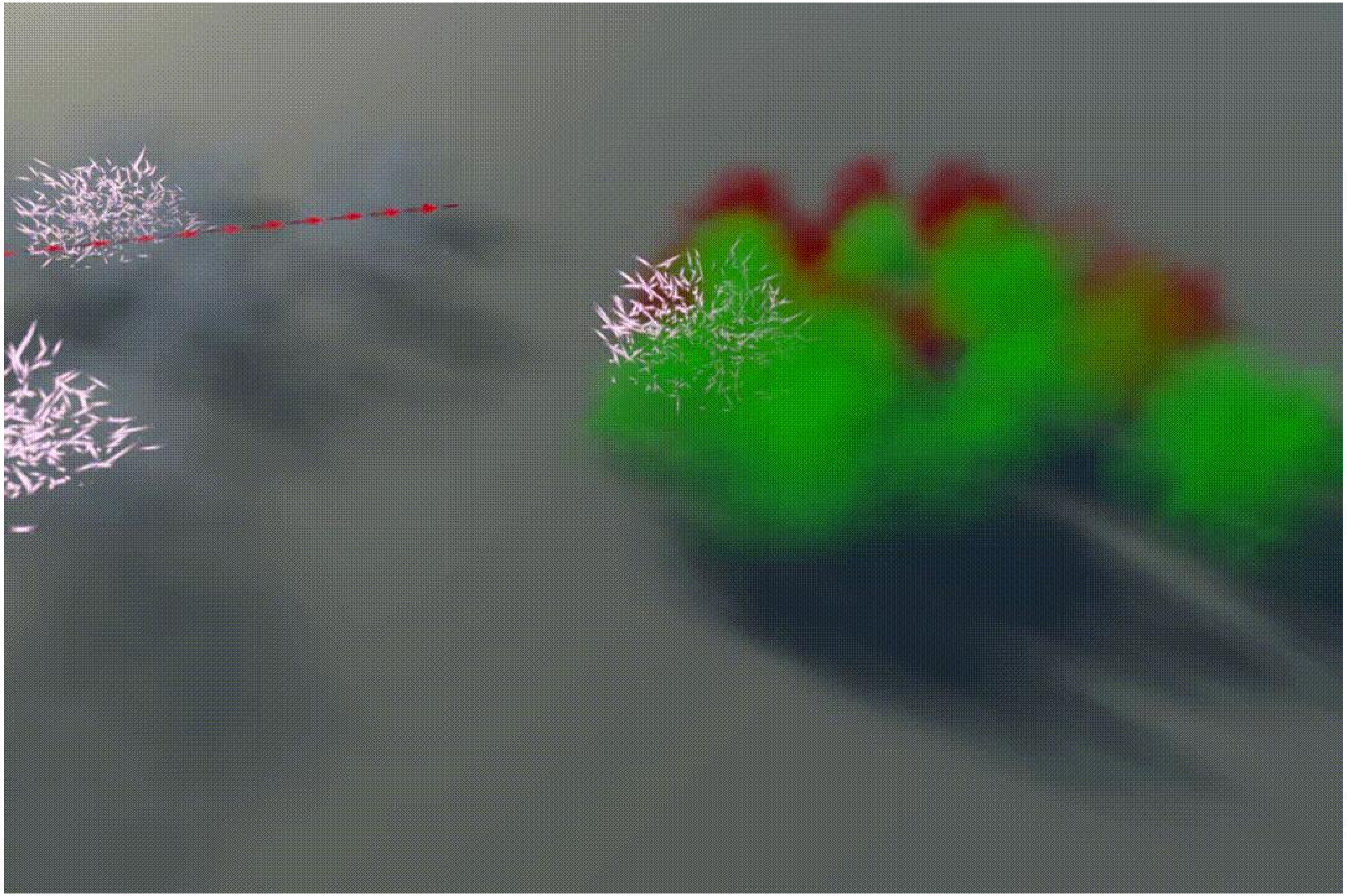


Global sort

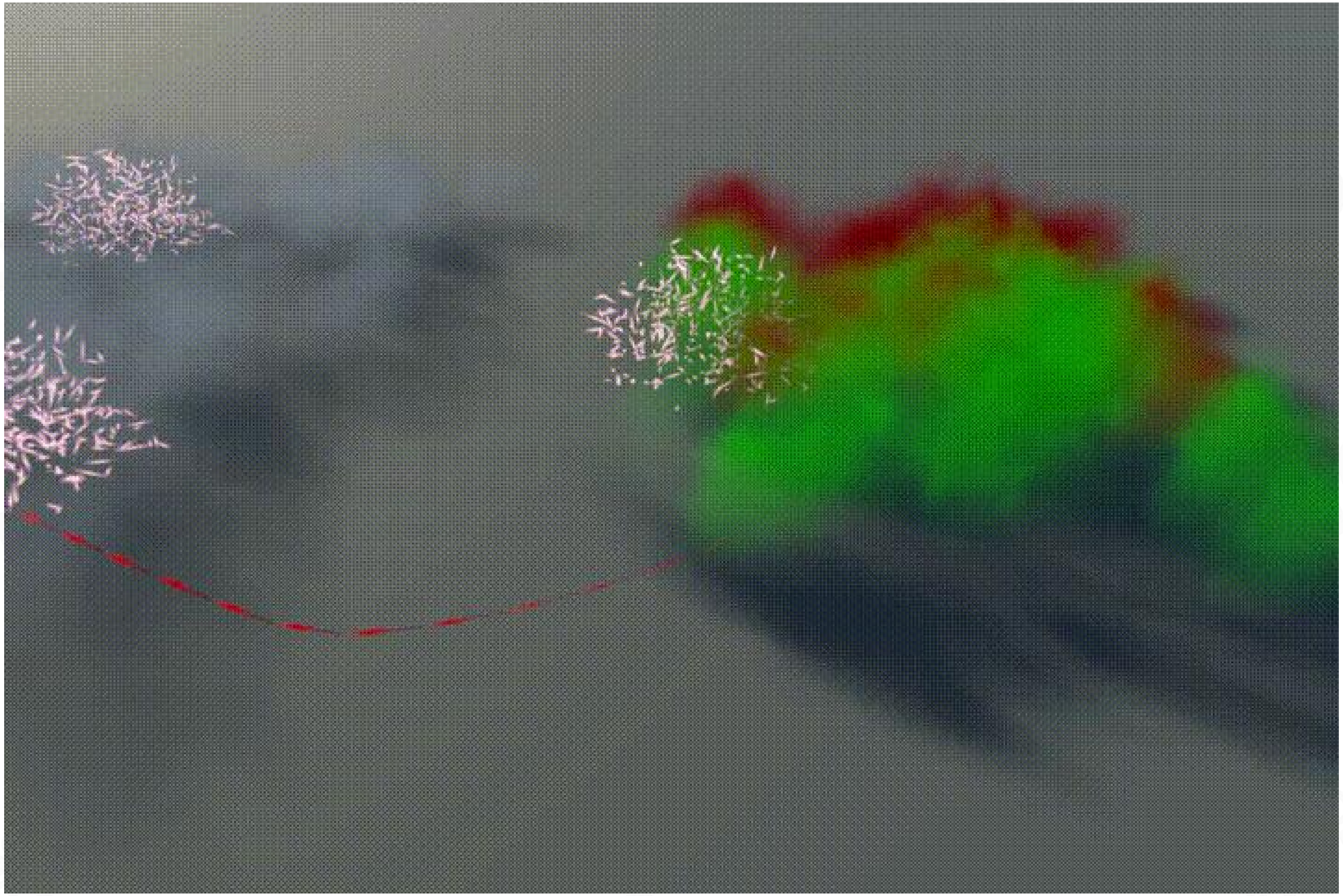


Per emitter sort

Per Emitter Sort



Global Sort



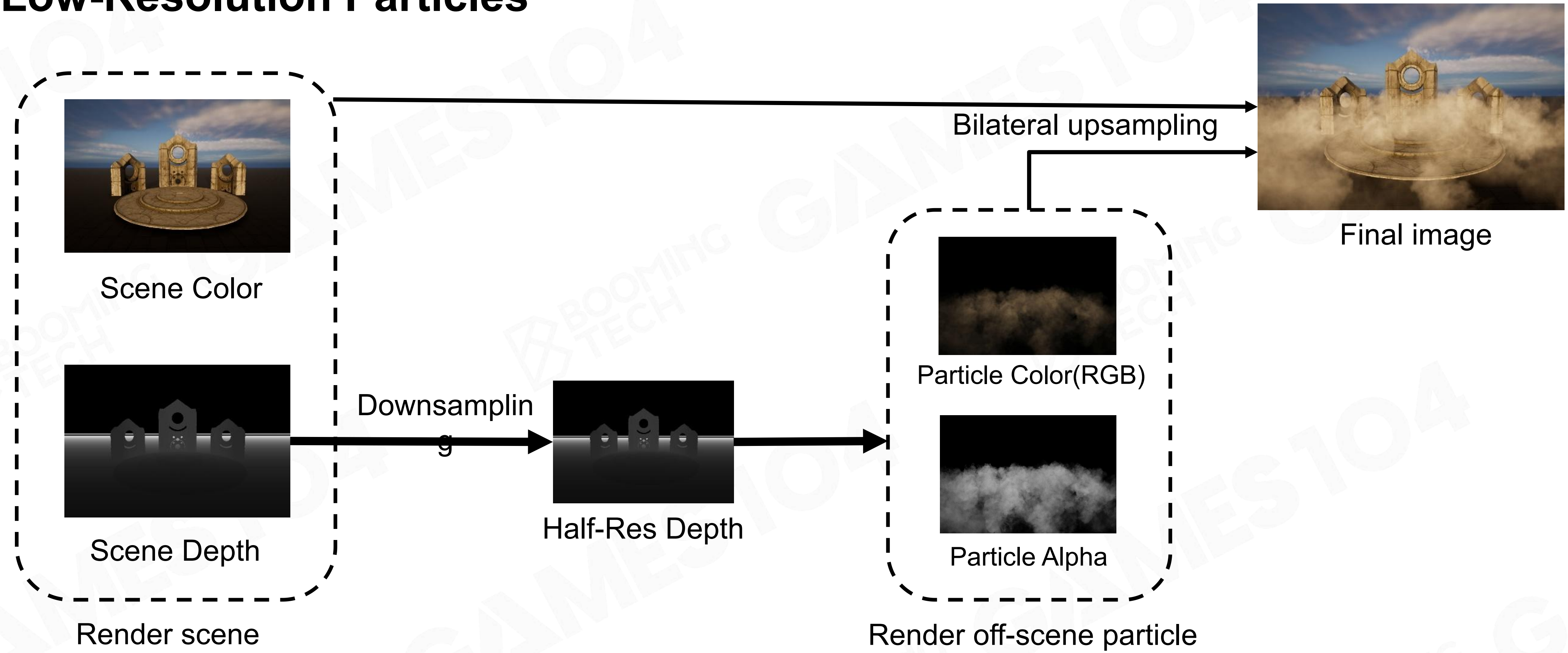
Full-Resolution Particles

draw opaque scene

render translucent particles

- Costy
- Worst case as particles fill the screen

Low-Resolution Particles



result_color = dst_color * (1-src_alpha) + src_color * src_alpha

result_alpha = dst_alpha * (1-src_alpha)

GPU Particle

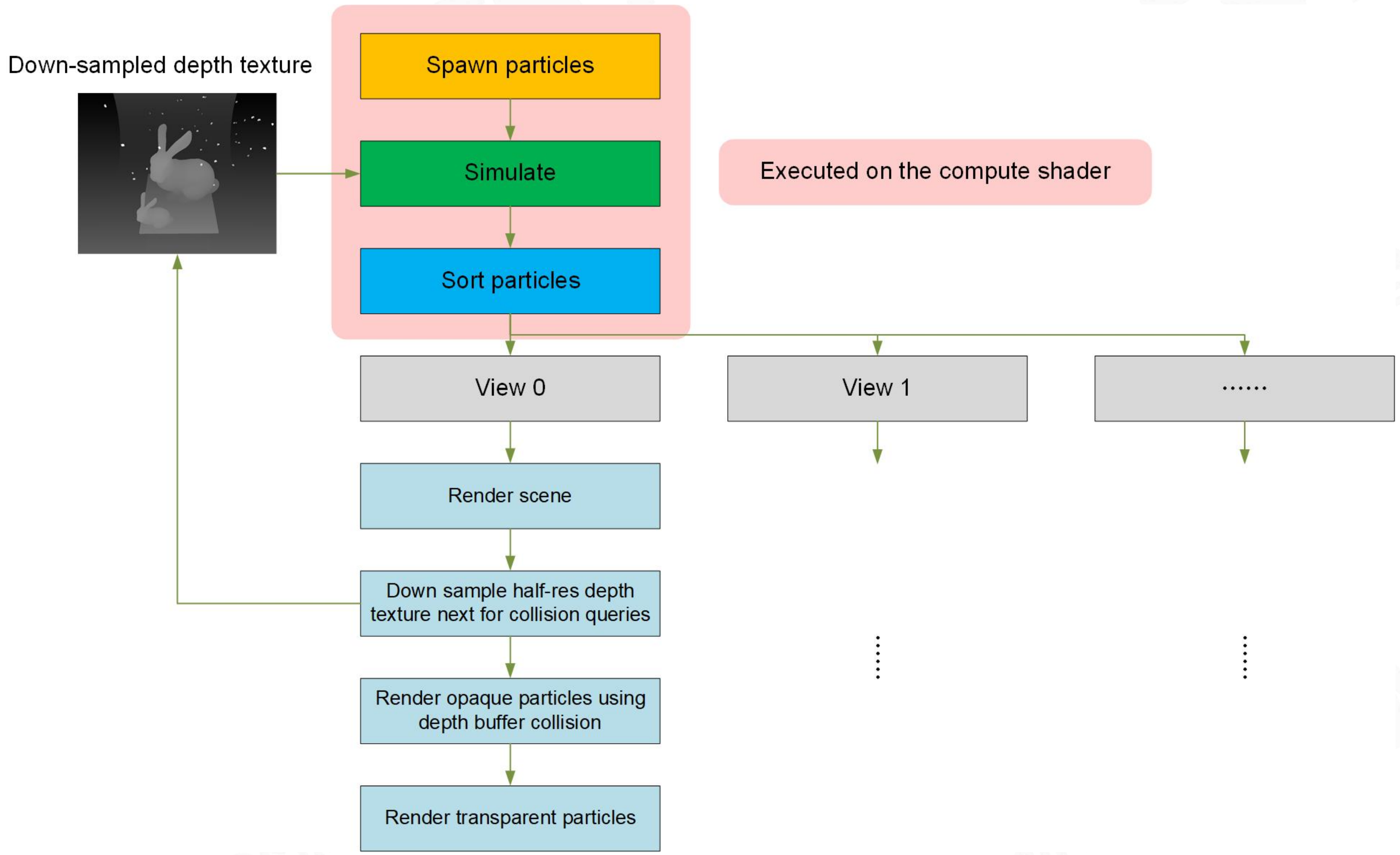
Processing Particles on GPU

Why use GPU?

- Highly parallel workload, suitable for simulation of large numbers of particles
- Free your CPU to do game code
- Easy to access depth buffer to do collision

GPU Particles - Frame Overview

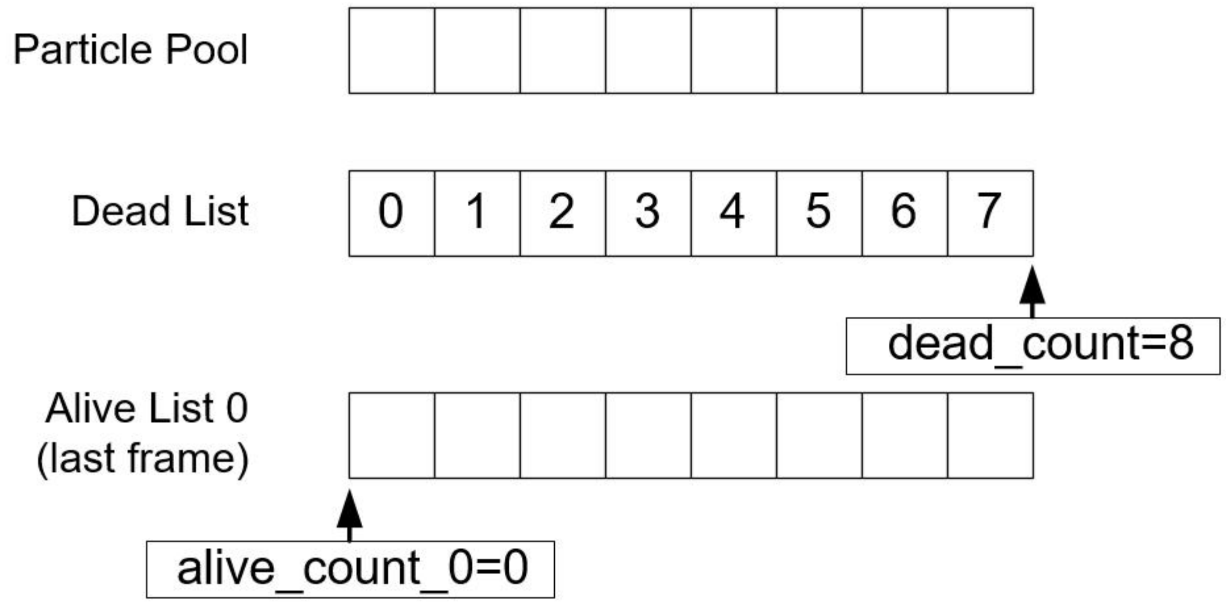
Compute shader provides high-speed general purpose computing and takes advantage of the large numbers of parallel processors on the GPU



Initial State

The diagram on the right shows an empty pool containing a maximum of 8 particles, starting with all 8 slots in a DEAD usable state

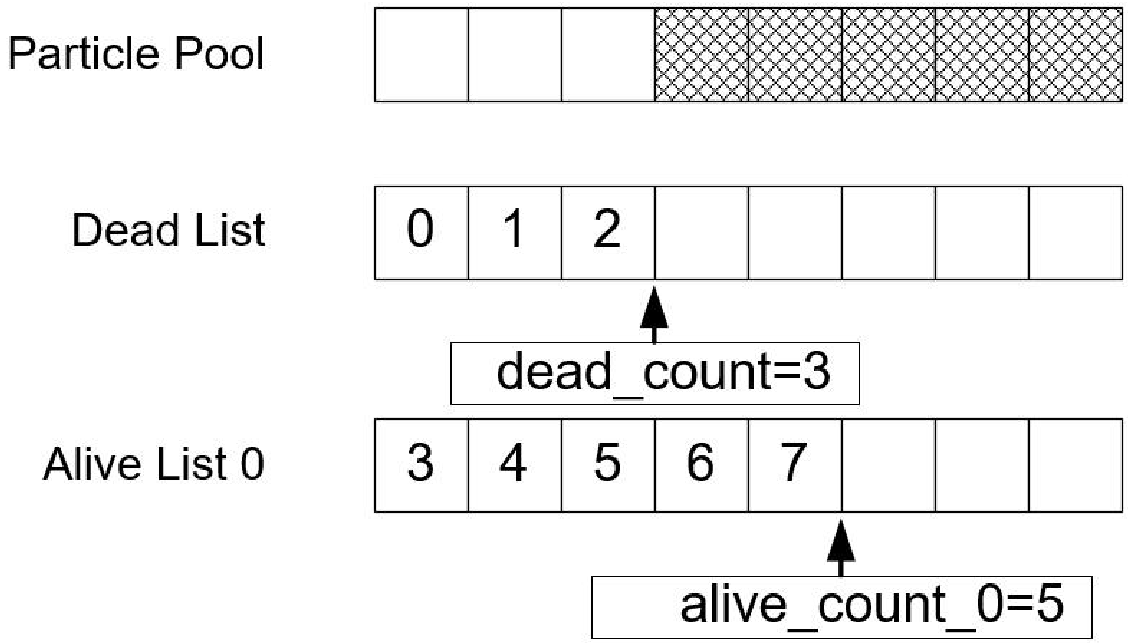
Particle pool is a single buffer storage containing all particles data



Spawn Particles

The diagram on the right shows the emission of 5 particles

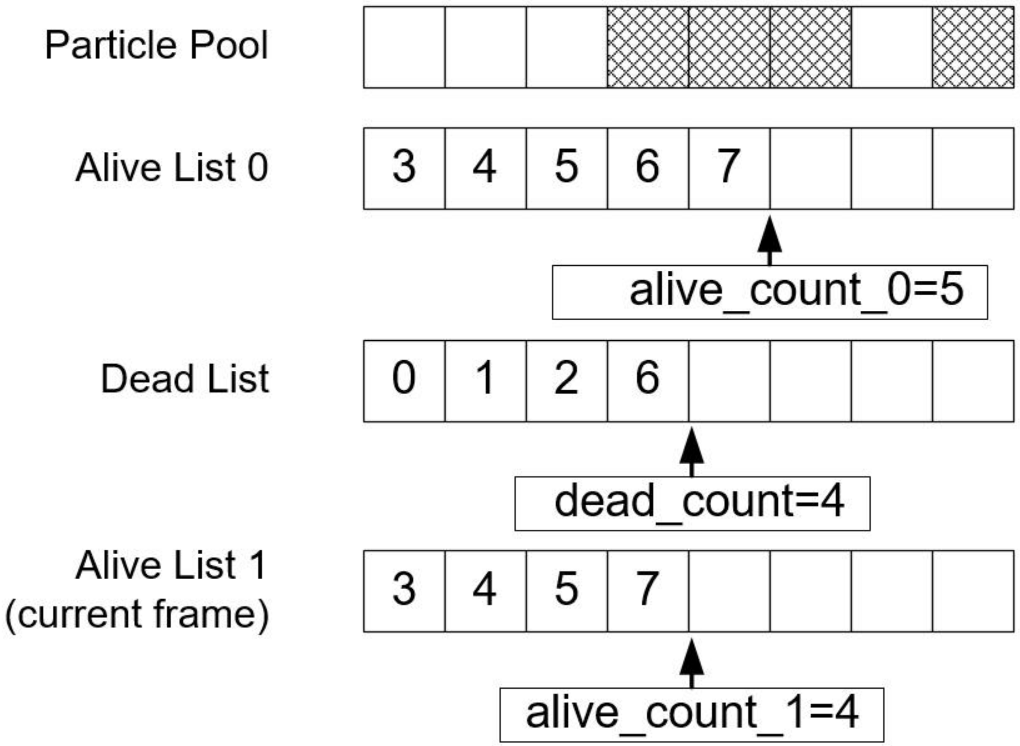
- Dispatch 5 compute shader threads to do the spawn calculation
- Access to the dead list and the alive list needs to be atomic



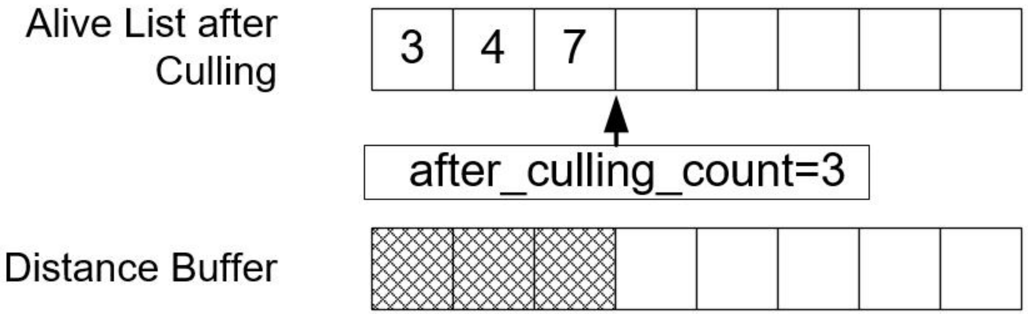
Simulate

Calculate position, velocity, doing depth collision, etc. And writing data back to particle pool
(right diagram shows if particle 6 is dead)

- Dispatch alive_count_0 threads
- Access to dead list and alive list 1 should be atomic

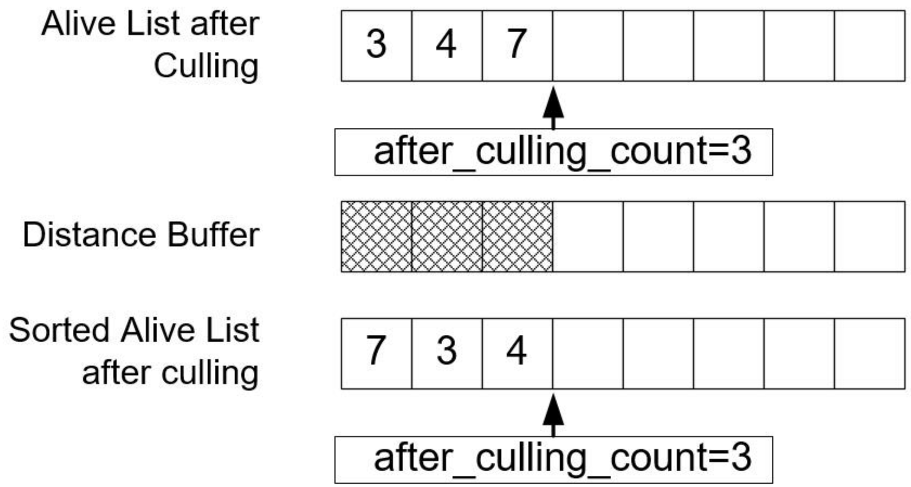


Do frustum culling, and write calculated distance to distance buffer
(right diagram shows if particle 5 is culled)

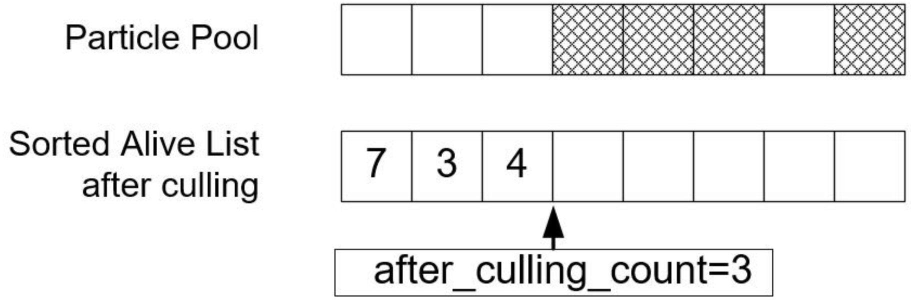


Sort, Render and Swap Alive Lists

Sort alive list after culling according to distance buffer



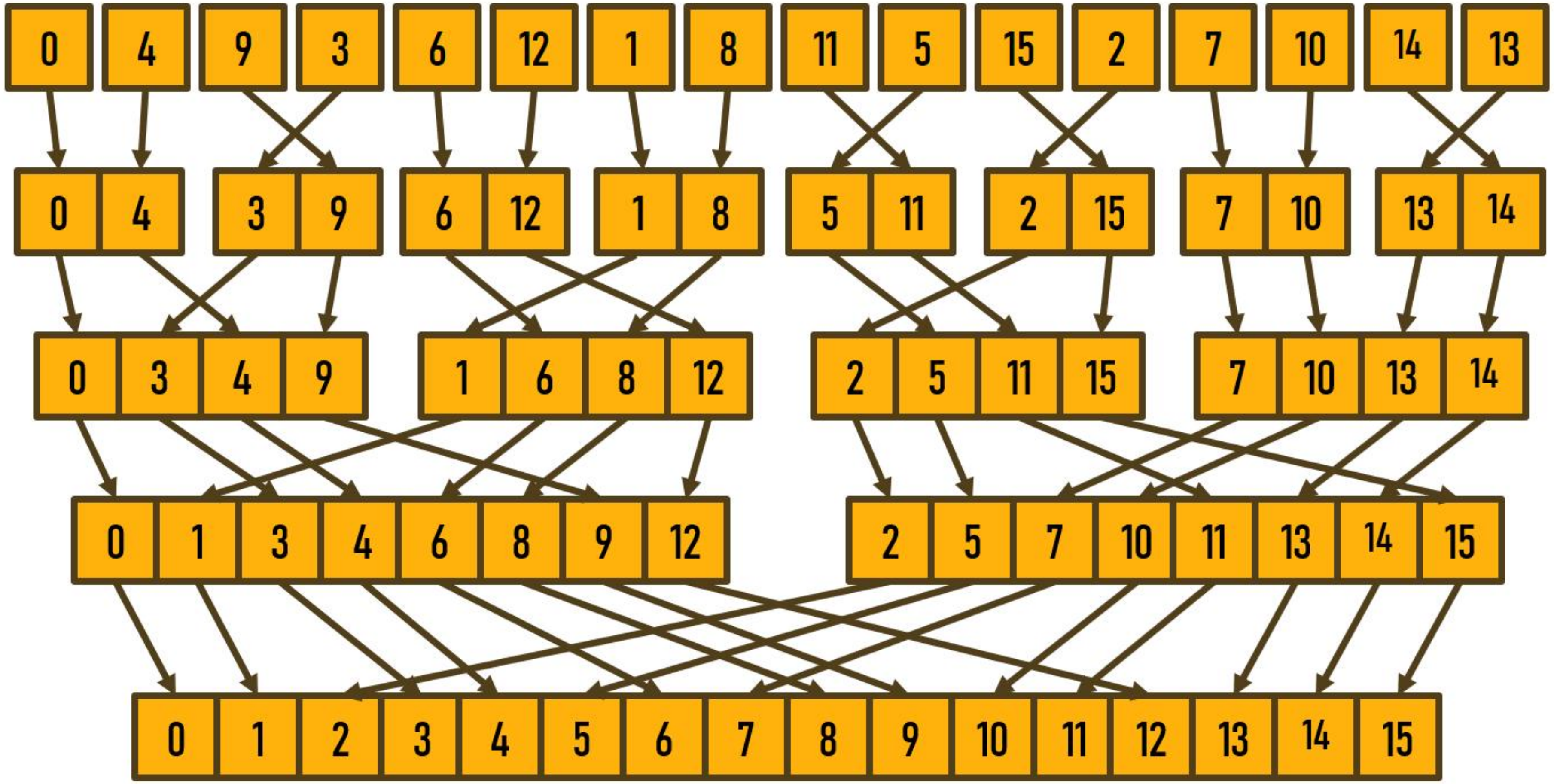
Render particles in sorted alive list after culling



Swap alive list

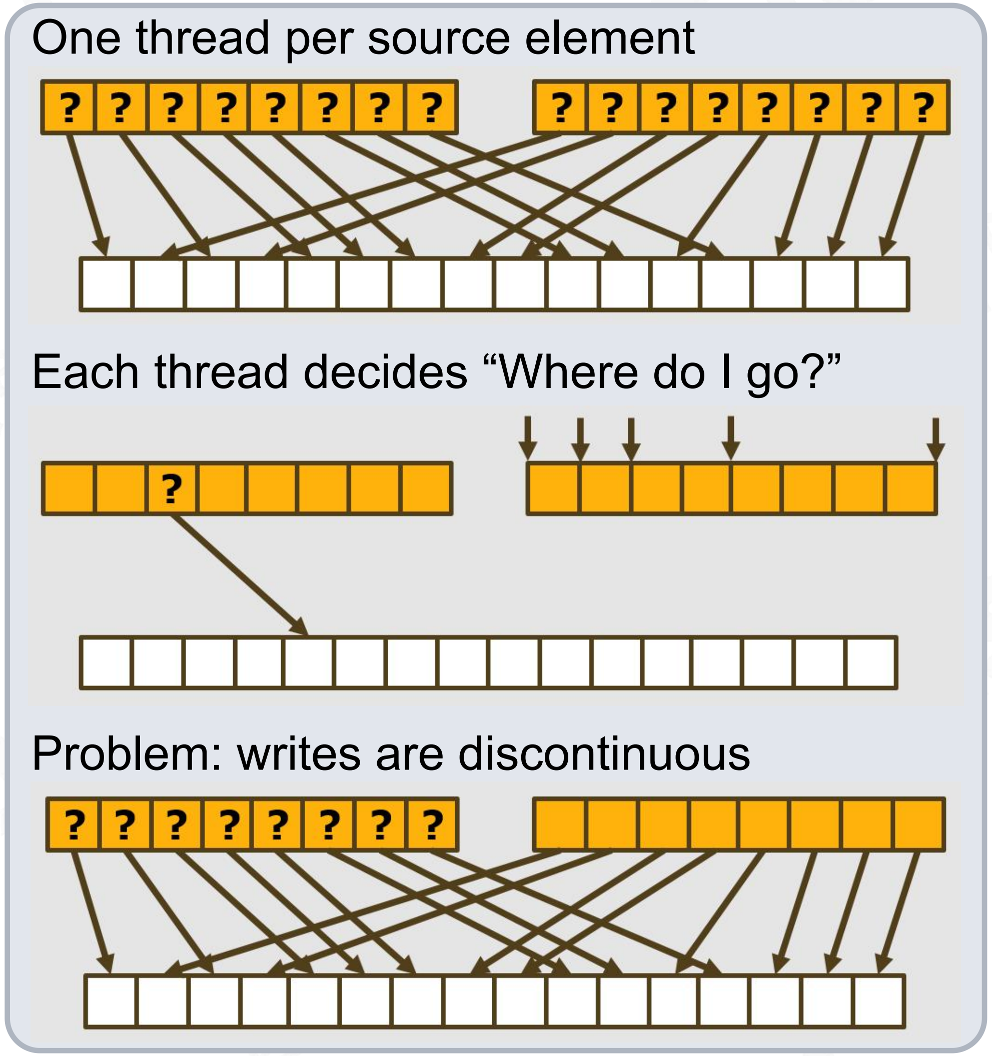
Swap Alive List 0 and Alive List 1
(as well as alive_count_0 and alive_count_1)

Parallel Mergesort

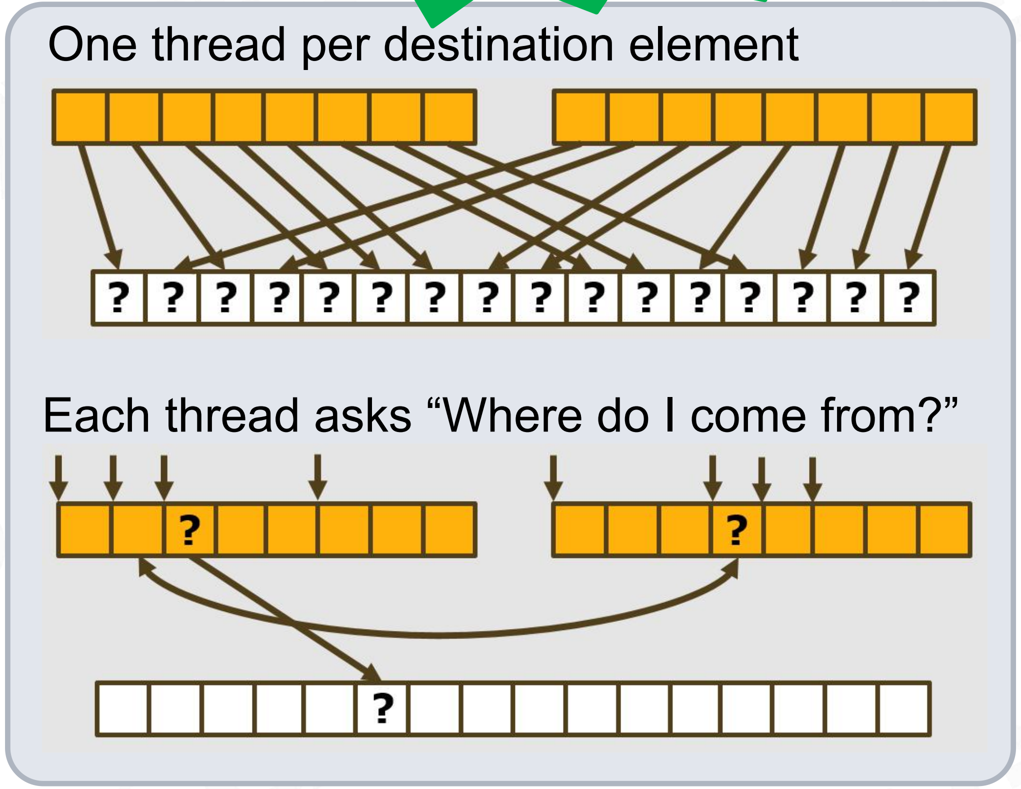


Parallel Mergesort

Method 1



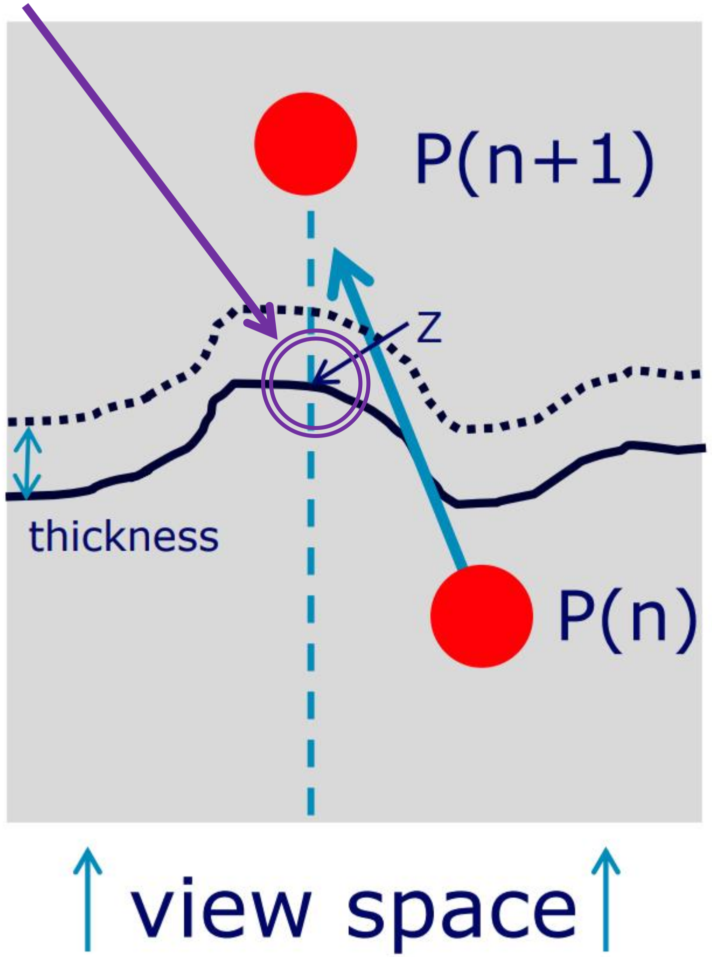
Method 2 (Better)



Depth Buffer Collision

1. Reproject particle position to previous frame screen space texture coordinate
2. Read depth value from presious frame depth texture
3. Check if particle is colliding with the depth buffer, but not completely behind it (where thickness value is used)
4. If collision is happened, calculate surface normal and bounce off the particle

Collision position

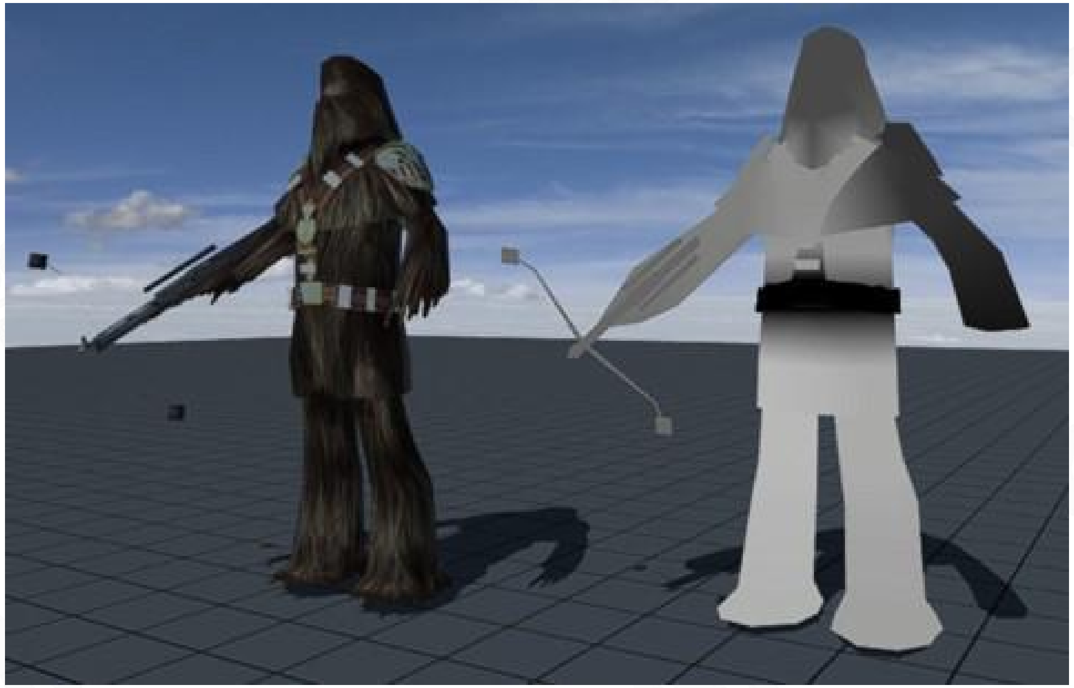


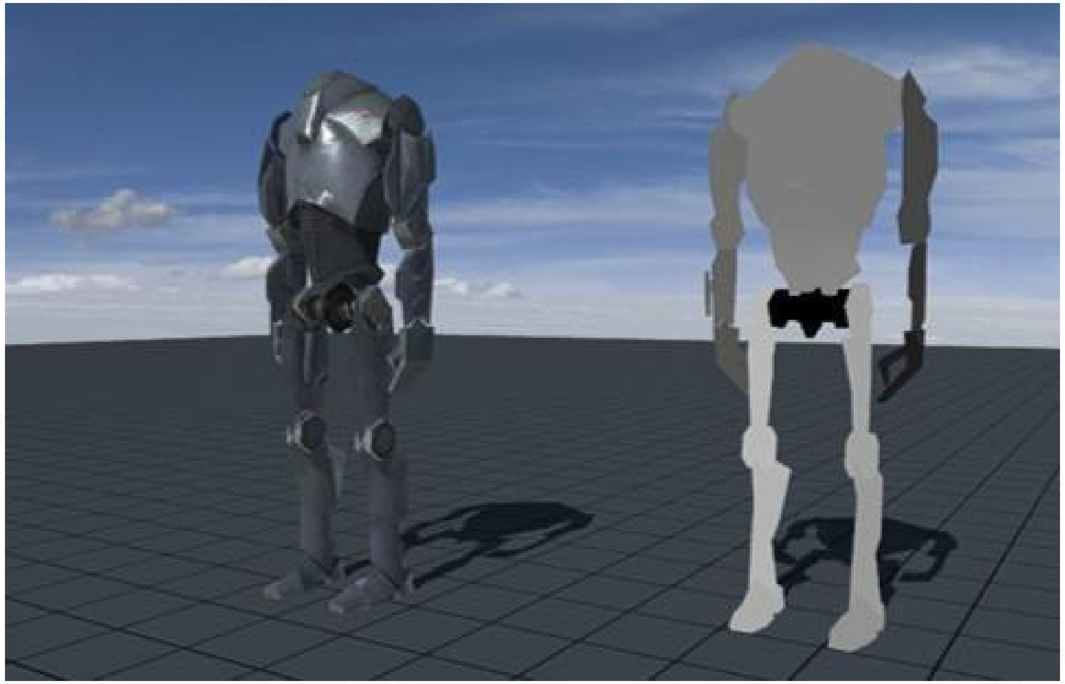
Collision Demo

Advanced Particles


Crowd Simulation

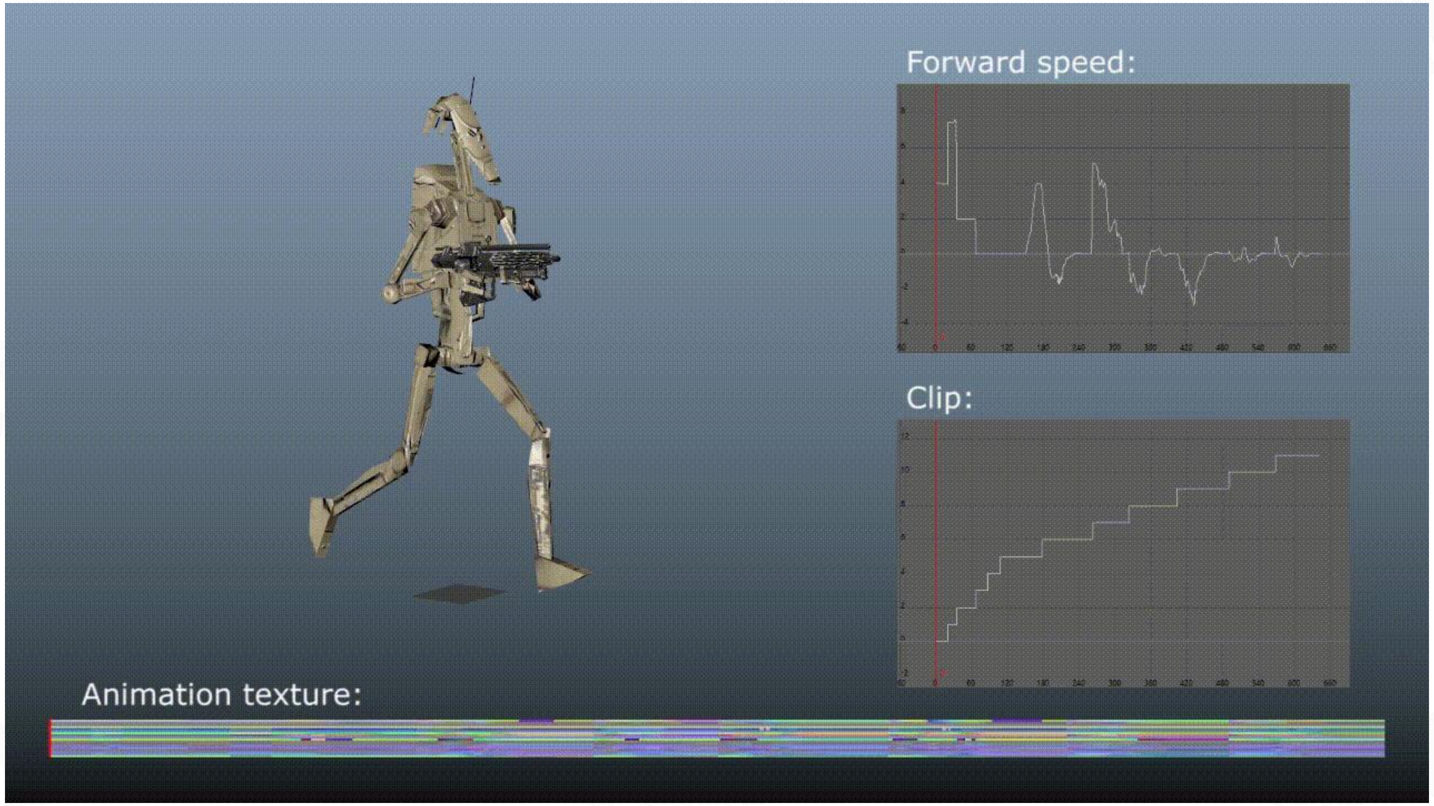
Animated Particle Mesh





Alpha (vertex_position.w) = Joint Index

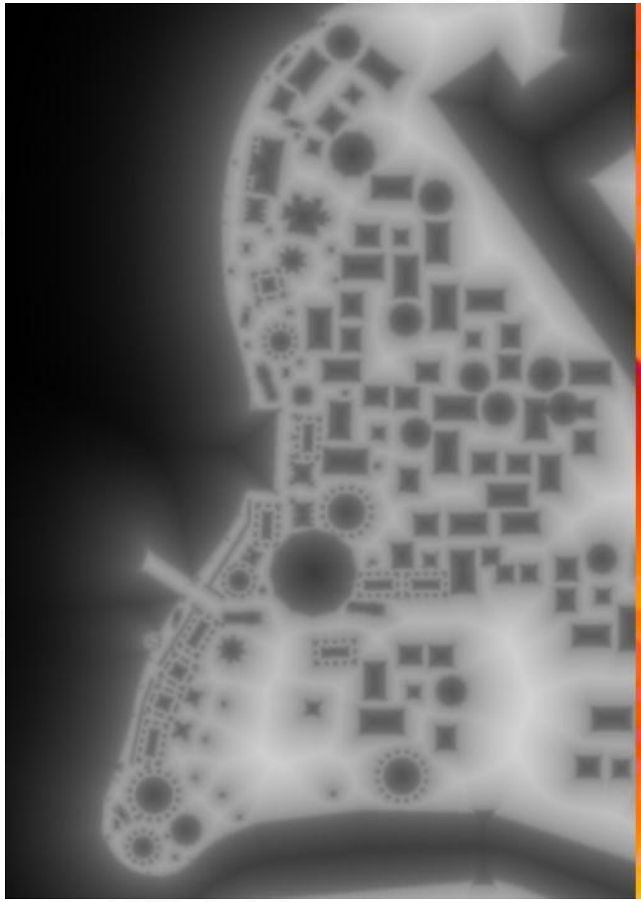
Particle Animation Texture




```
#define CLIP_run 0
#define CLIP_sprint 1
#define CLIP_walk 2
#define CLIP_standFire 3
#define CLIP_crouchFire 4
#define CLIP_command 5
#define CLIP_deathRun 6
#define CLIP_deathSprint 7
#define CLIP_deathStandA 8
#define CLIP_deathStandB 9
#define CLIP_deathCrouchA 10
#define CLIP_deathCrouchB 11

const int numClips = 12;
const int clipStarts [12] = {0, 20, 35, 67, 87, 107,
const int clipLengths [12] = {20, 15, 32, 20, 20, 70
const float clipAvgSpeeds [12] = {3.99178, 7.54395,
const float maxSpeed = 7.63941650391;
const int numFrames = 643;
const int numJoints = 17;
const float3 jointOffsets [17] = {float3(0.0, 0.0, 0
```

Navigation Texture

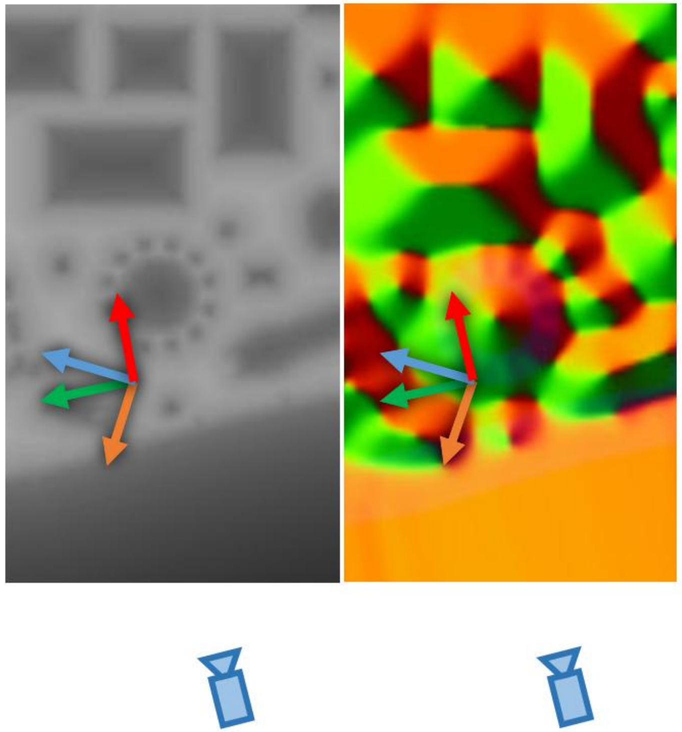


Signed Distance Field

Direction Texture (RG channels)

Crowds - Runtime Behavior

- Design target locations to guide the movement of crowds
- The desire moving towards the target location, pushing away from blocking geometry, all become forces to influence the movement of crowds (if close enough, the camera also acts as a force)



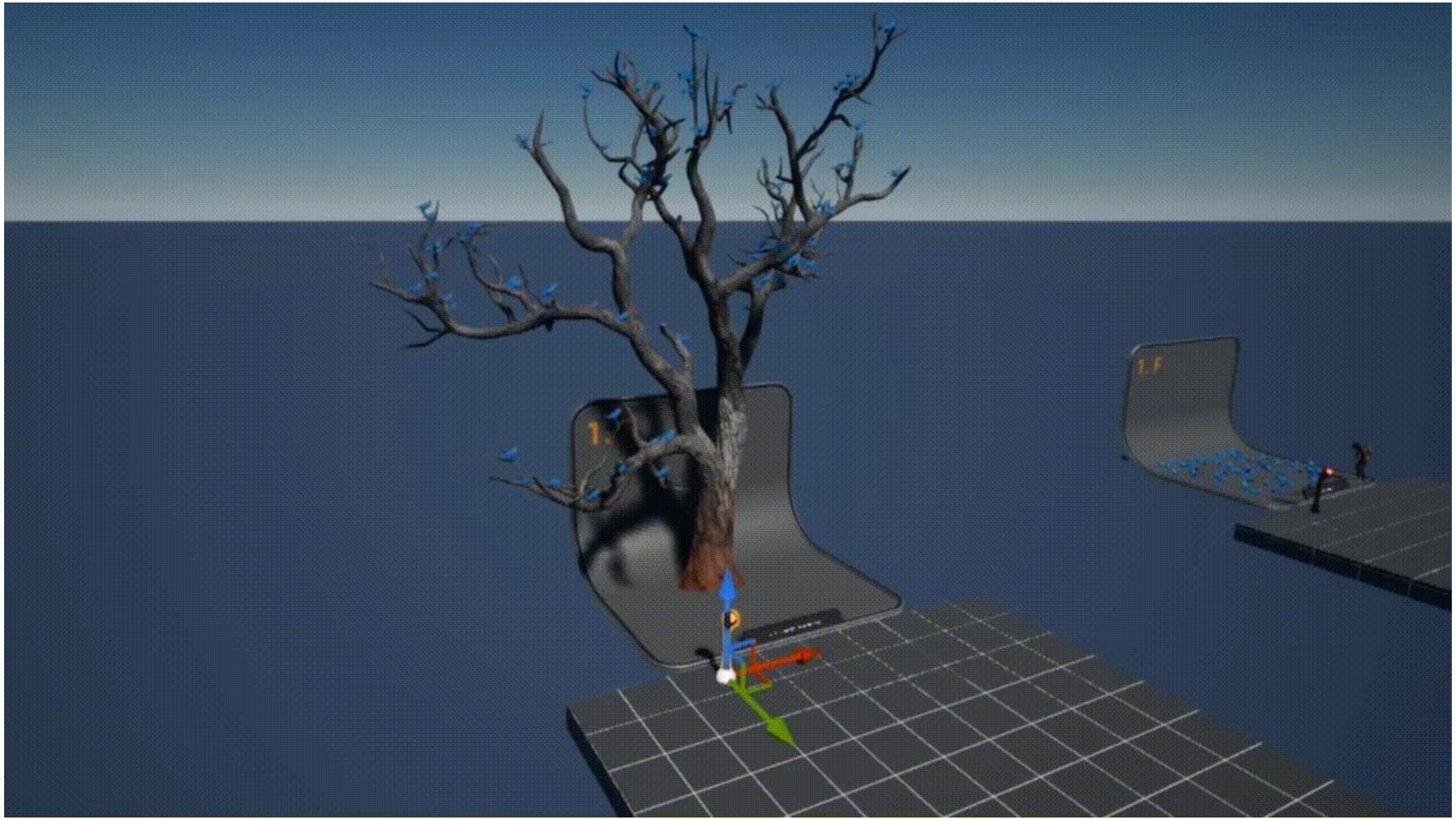
Advanced Particle Demos (1/2)

- Skeleton mesh emitter
- Dynamic procedural splines
- Reactions to other players

……

Advanced Particle Demos (2/2)

- Interacting with environment



Utilizing Particle System in Games

Design Philosophy - Preset Stack-Style Modules

Pros

- Fast to add behaviors as stacked modules
- Non technical artists have lots of control via a suite of typical behaviors
- Easy to understand at a glance

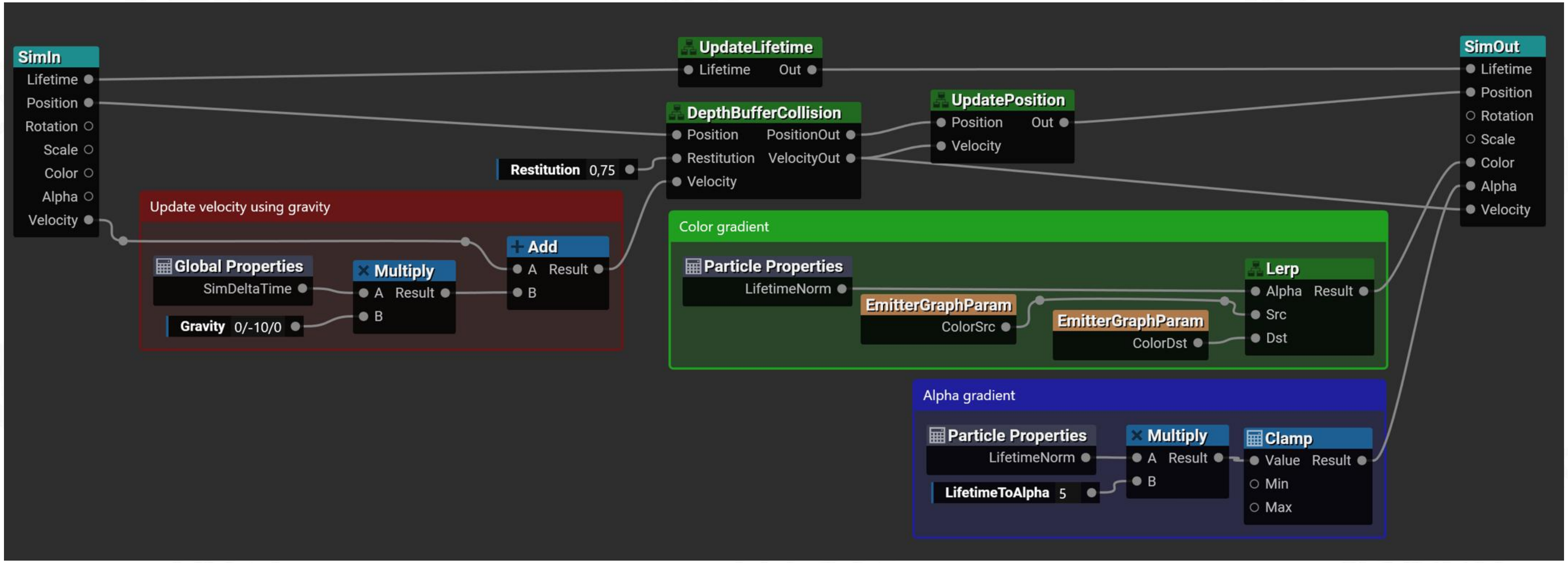
Cons

- Fixed functions, new feature requires new code
- Code-based, divergence in game team code
- Fixed particle data, data sharing is mostly impossible

Cascade Particle System in Unreal Engine

Design Philosophy - Graph-Based Design

- Parameterizable and shareable graphs asset
- Less code divergence
- Provide modular tools instead of hardcoded features



Hybrid Design

- Graphs give total control
- Stacks provide modular behavior and glance readability

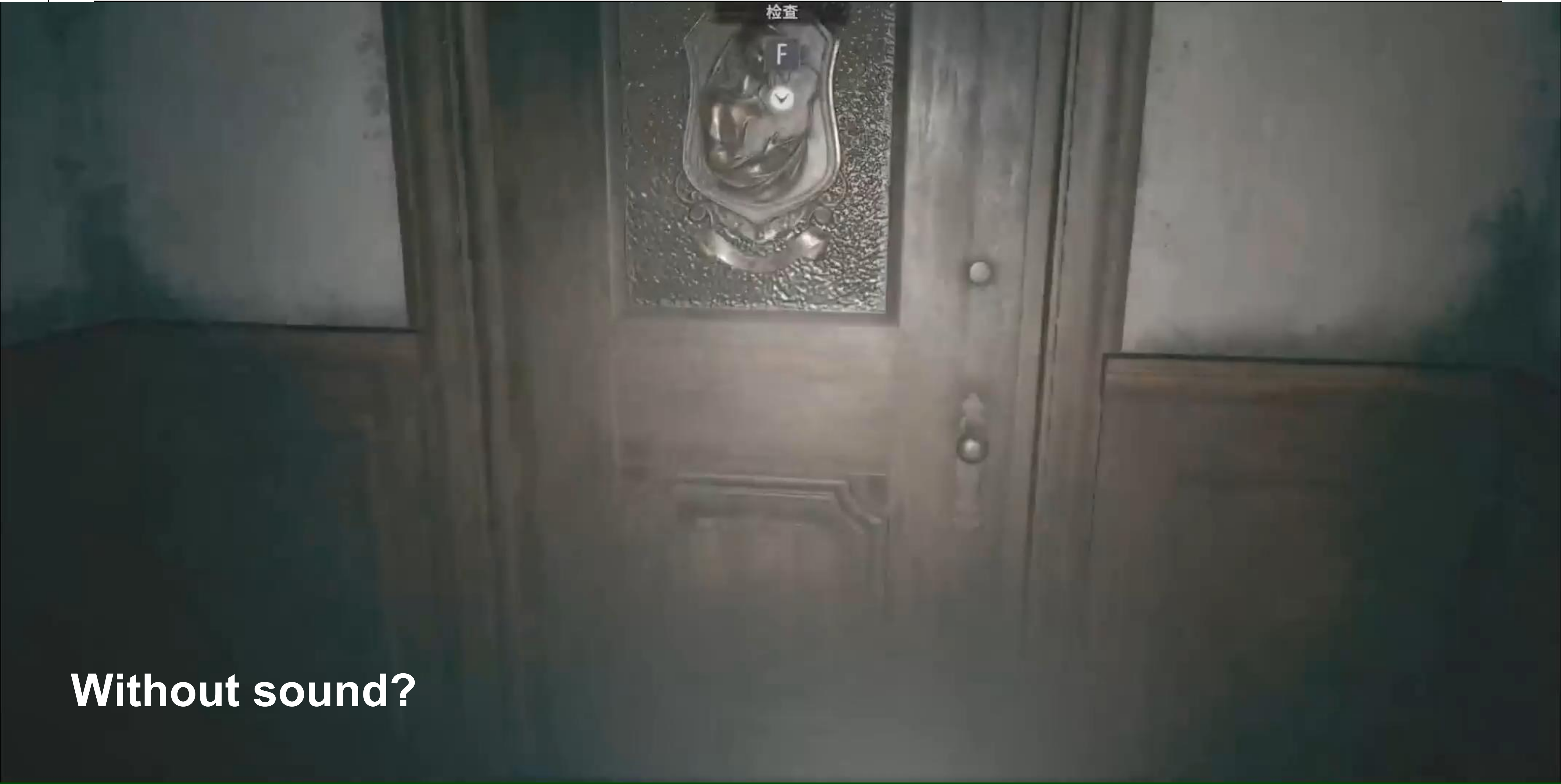
Modules	Emitters	Systems
• Encapsulate behaviors (seen as "write functions") • Stack with each other	• Containers for modules • Single purpose, reusable	• Holders of multiple emitters into one "Effect"
Graph	***Stack***	***Stack***

Unreal's Niagara System Design

Sound System

Audio

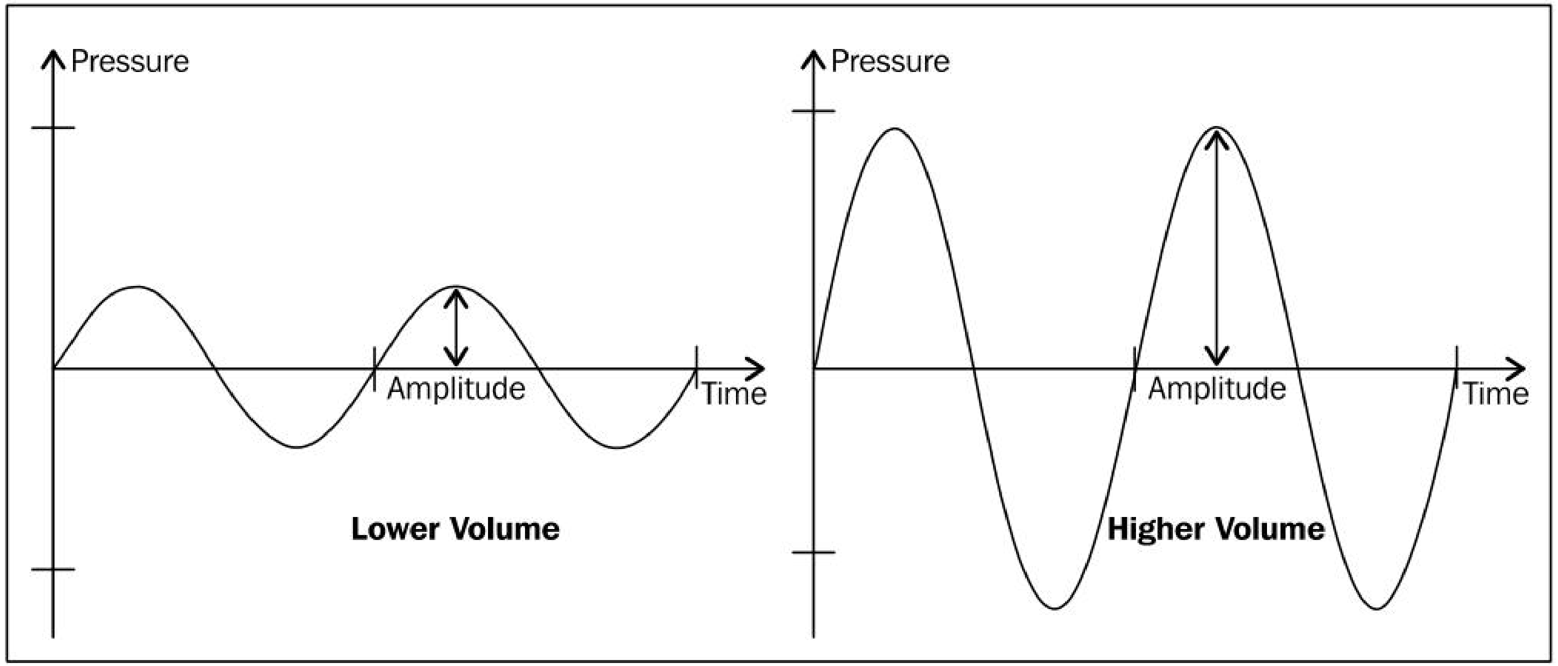
- Entertains the player
- Enhances the realism
- Establishes atmosphere

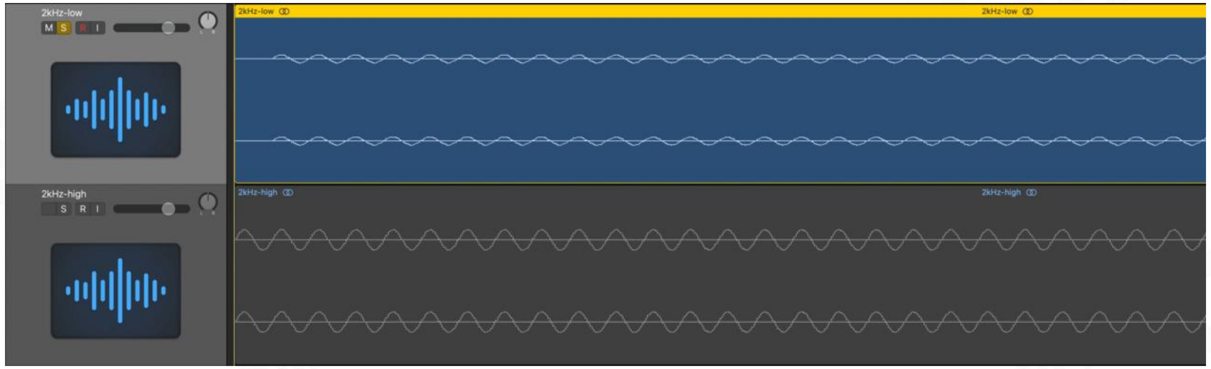

检查
F
Without sound?


检查
F
With Sound

Volume

The amplitude of the sound wave





Volume - Terminologies (1/2)

Sound Pressure ([illegible]): Local deviation from the ambient atmospheric pressure caused by sound wave, SI unit ([illegible])

Particle Velocity ($s_1 a_1$): The velocity of a particle in a medium as it transmits a wave, SI unit ($1 - a_1$)

Sound Intensity ([illegible]): The power carried by sound waves per unit area in a direction perpendicular to that area, SI unit: $RGB_0 = 0, A_0 = 1$

$$RGB_2 = s_2 a_2 + RGB_1 (1 - a_2), A_2 = (1 - a_2) A_1$$

Volume - Terminologies (2/2)

Sound Pressure Level (L_p): A logarithmic measure of the effective pressure of a sound relative to a reference value, SI unit ($(1-a_1)(1-a_2)$)

$$s_2 a_2 + s_1 a_1 (1 - a_2)$$

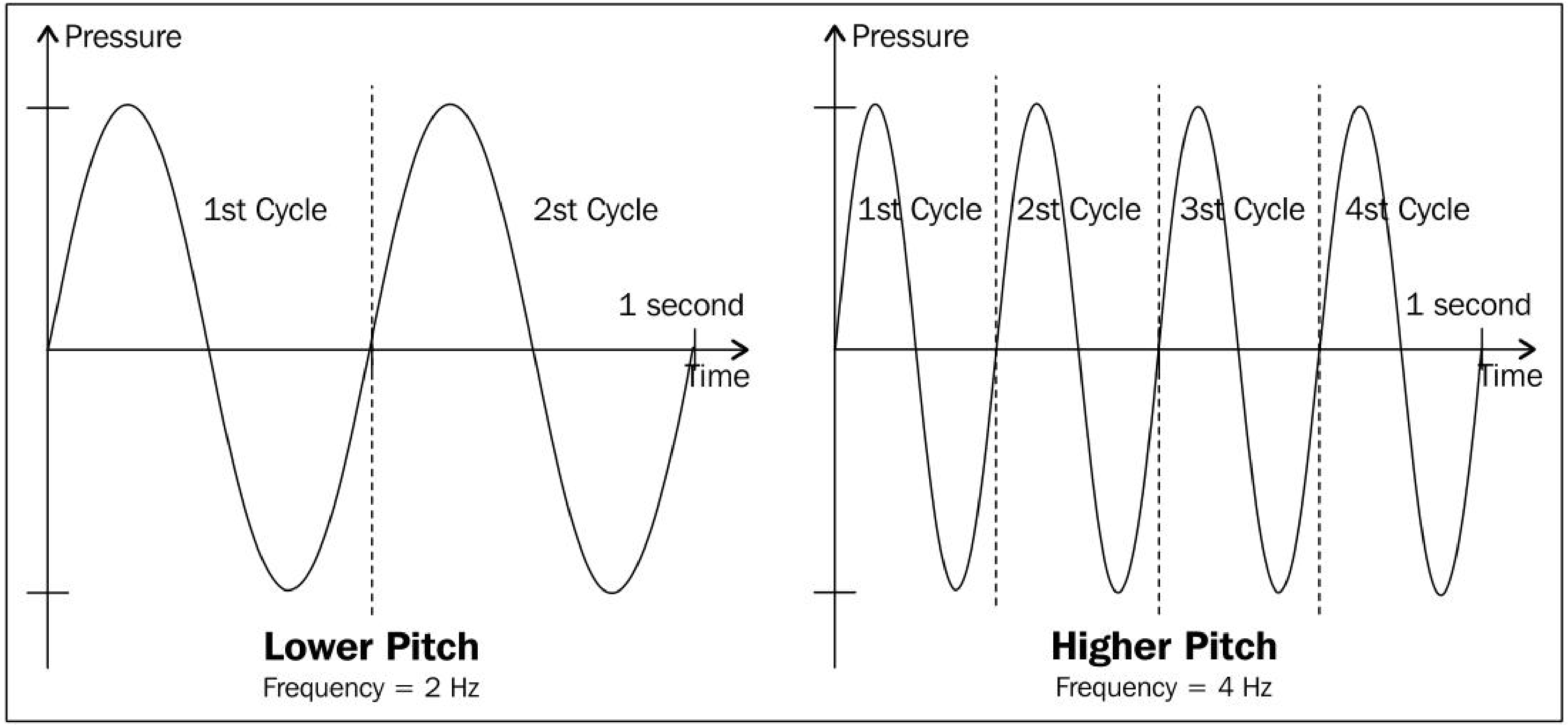
p_0 **:** Reference sound pressure, the threshold of human hearing, commonly used in air is

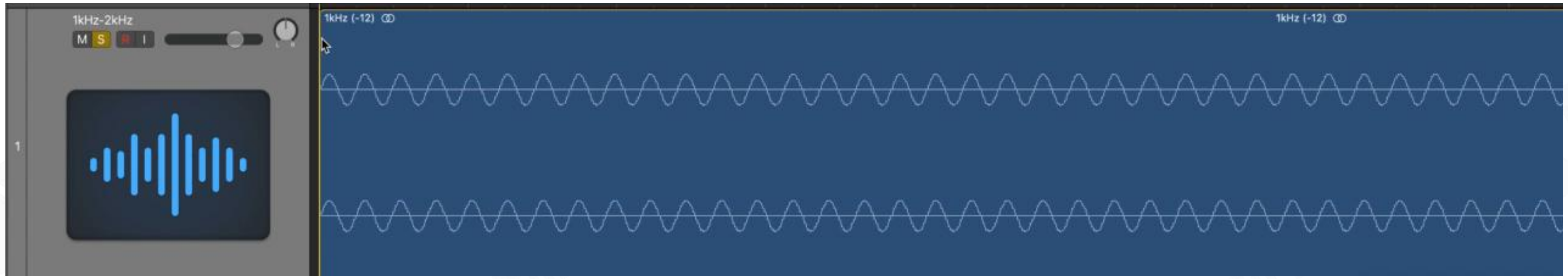
$$p_0 = 20\mu Pa$$

(roughly the sound of a mosquito flying 3 m away)

Pitch

- Determines how high or low the sound is
- Depends on the frequency of the sound wave

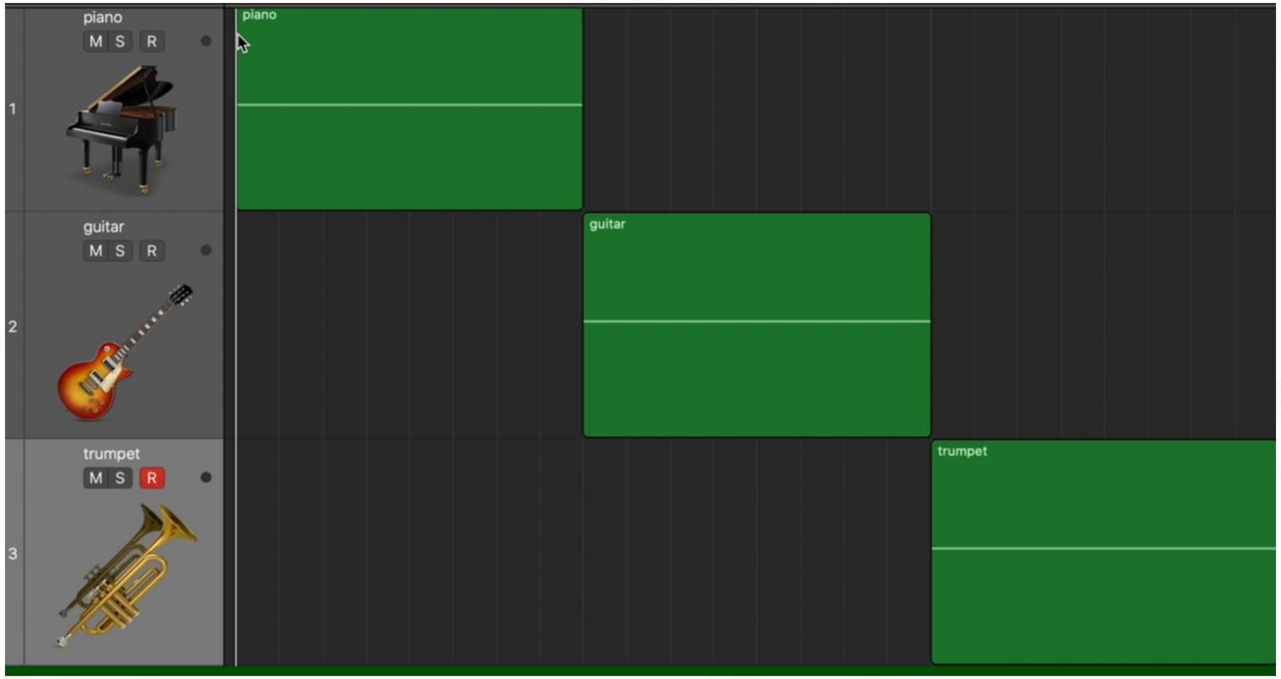




Timbre

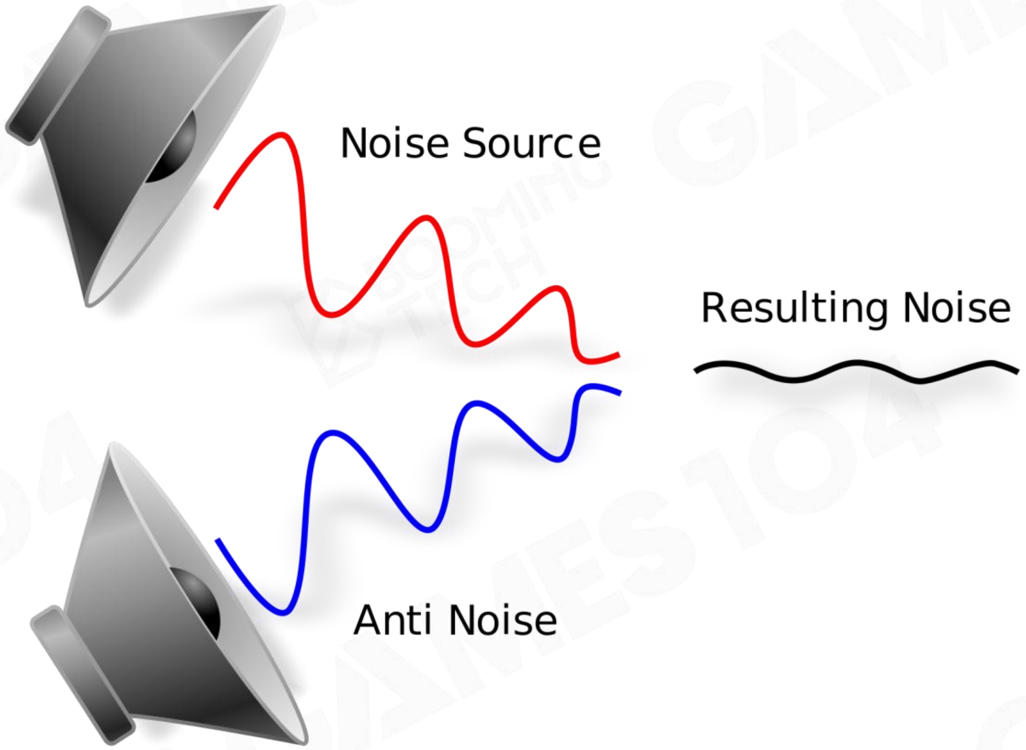
Combinations of overtones or harmonics

- frequencies
- relative intensities



Phase and Noise Cancelling

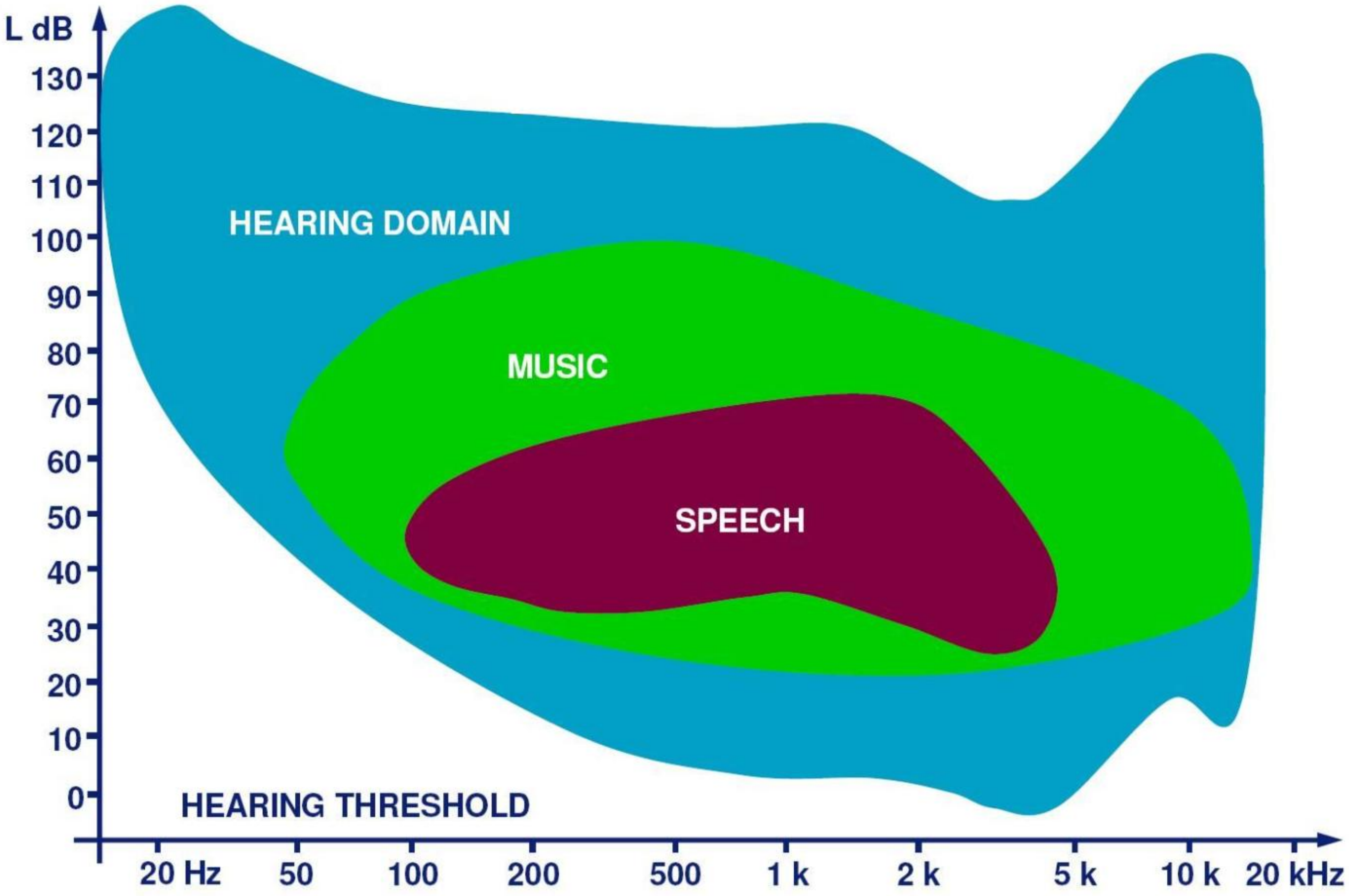
Same frequency, amplitude, but in different phase



Human Hearing Characteristic

The audible sound of human ear

- frequency range: 20-20KHz
- sound pressure level range (0-130db)

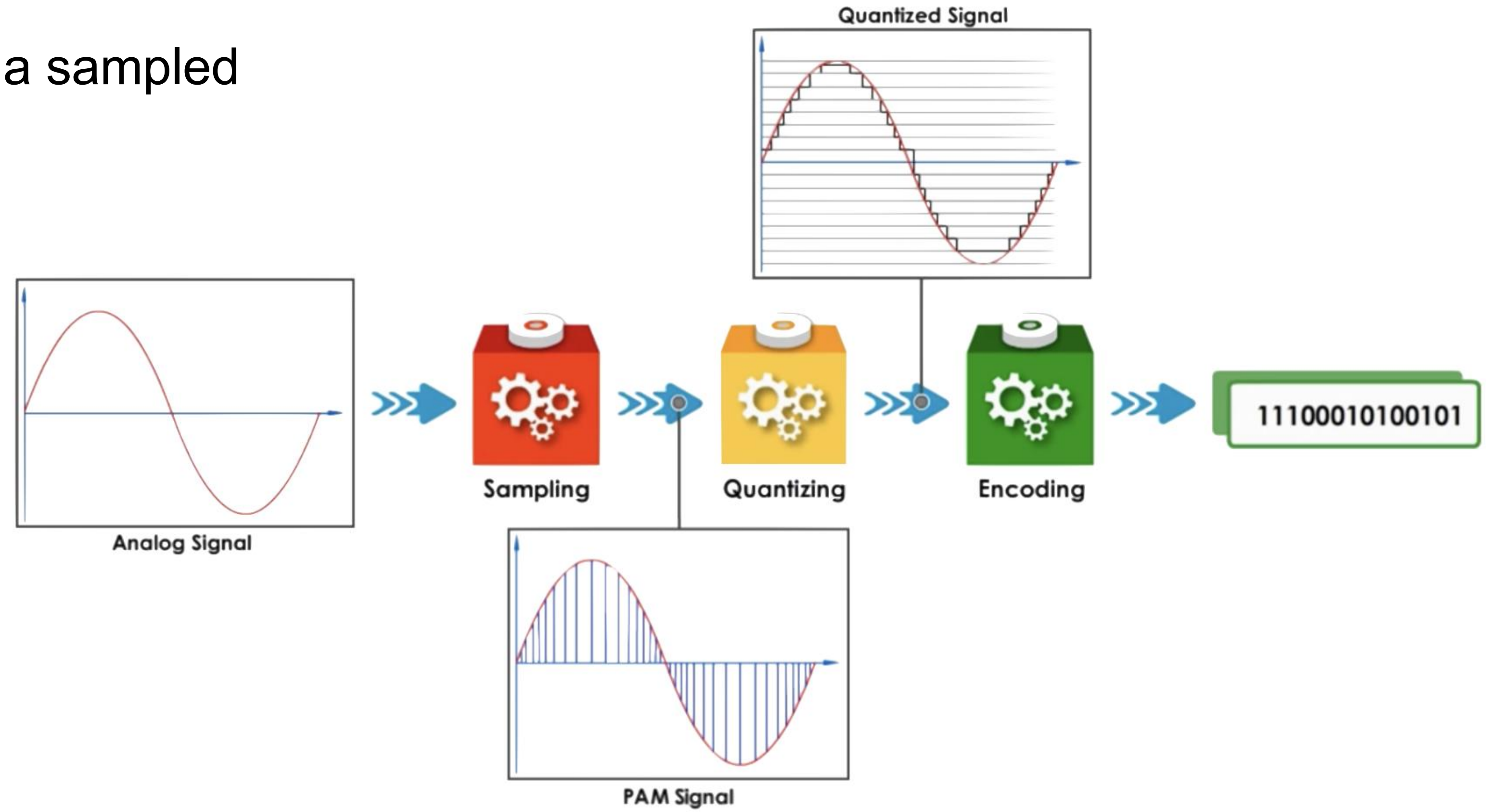


Digital Sound

Pulse-code Modulation (PCM)

Standard method for encoding a sampled analog sound signal

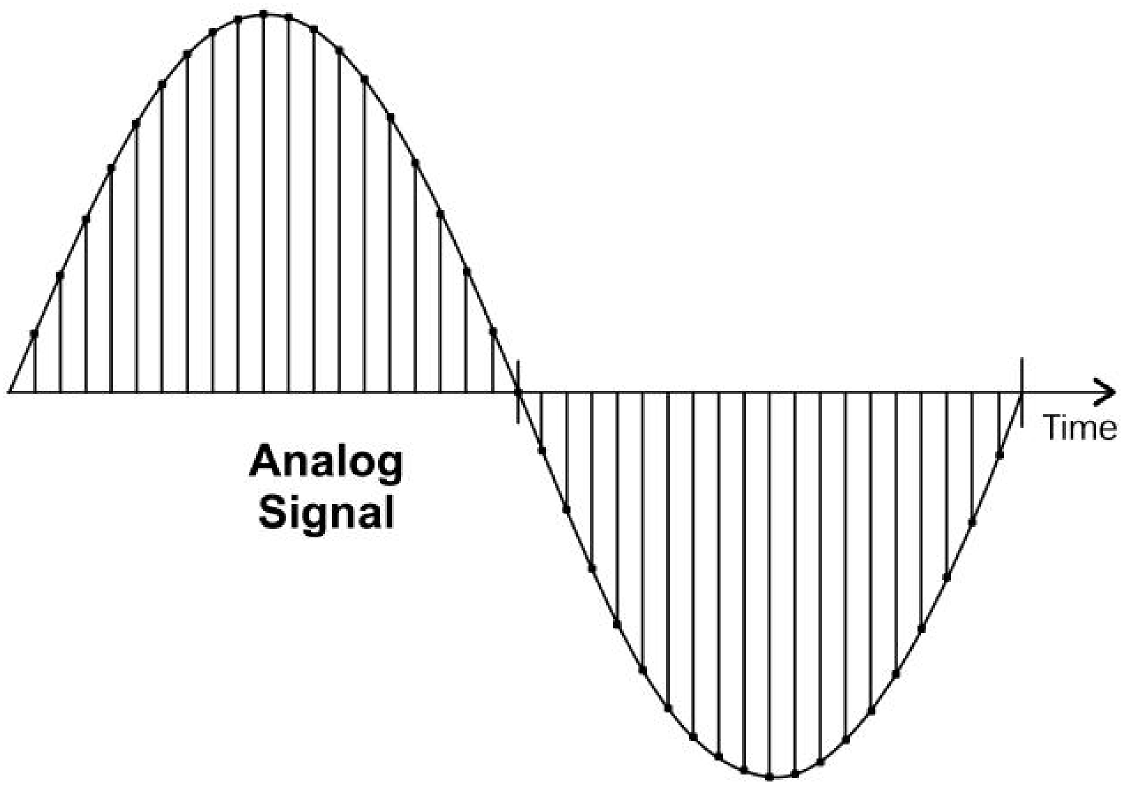
- Sampling
- Quantizing
- Encoding



Sampling

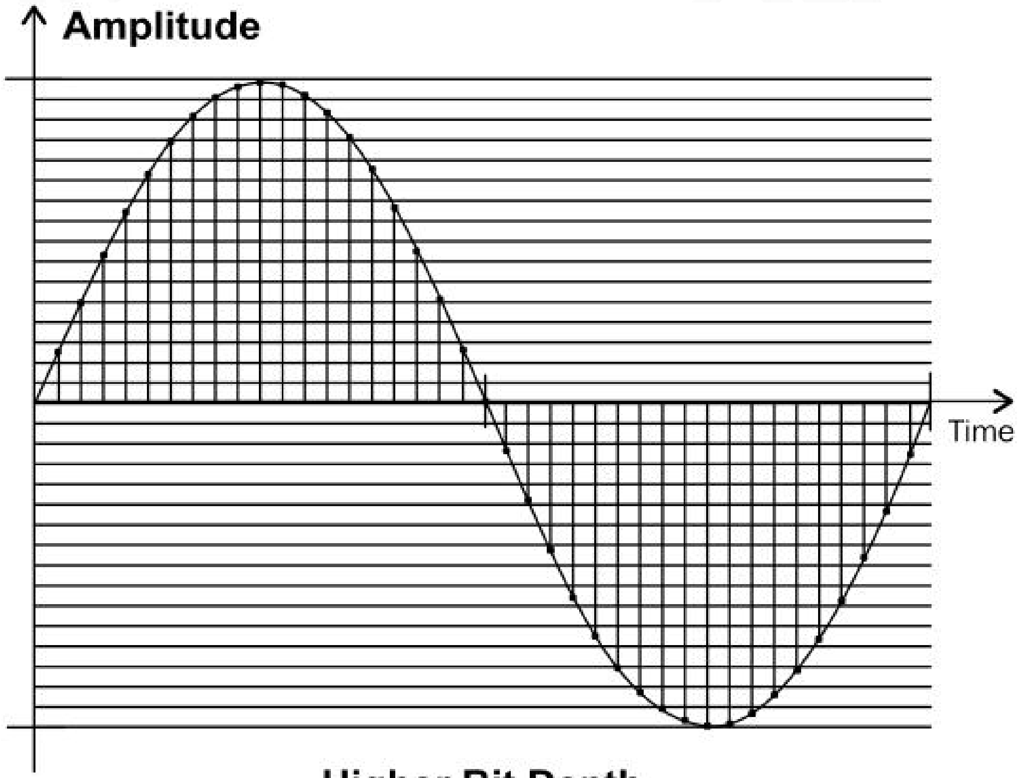
Sample Frequency：Samples per second(Hz)

Nyquist–Shannon Sampling Theorem: *The minimum sampling frequency of a signal that it will not distort its underlying information, should be double the frequency of its highest frequency component*

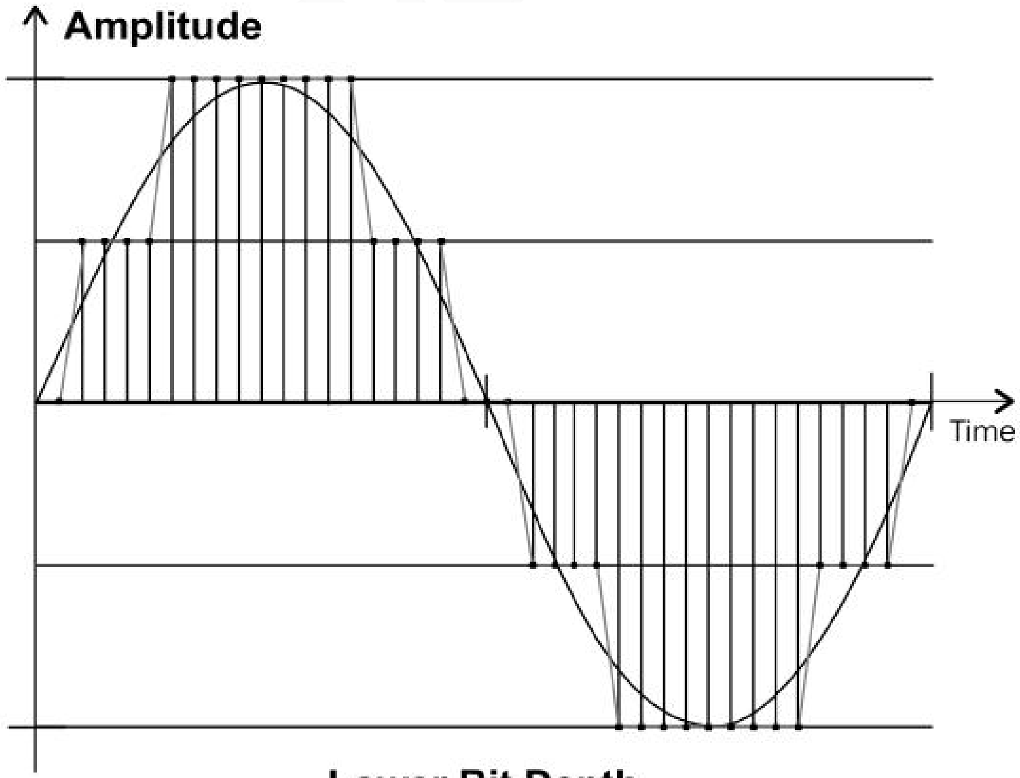


Quantizing

bit-depth：bit depth is the number of bits of information in each sample



Higher Bit Depth



Lower Bit Depth

Audio Format

Format	Quality	Storage	Multi-channel	Patent
WAV(uncompressed)	★★★	★	★★★	★★★
FLAC(lossless)	★★★	★★	★★★	★★★
MP3(lossy)	★	★★★	★	★
OGG(lossy)	★	★★★	★★★	★★★

3D Audio Rendering

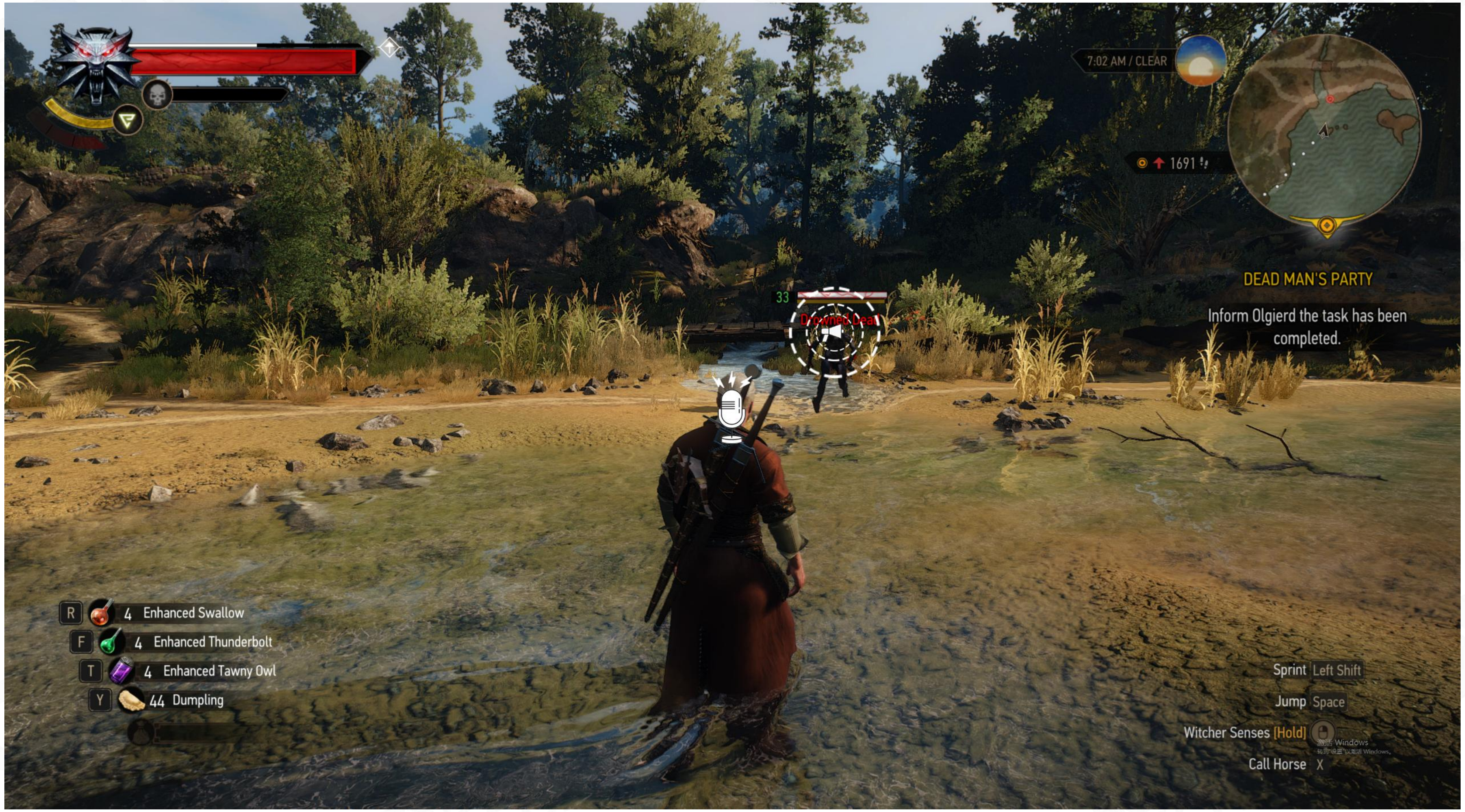
3D Sound Sources

- a mono-phonic audio signal
- emanating from a specific position

Listener

A "virtual microphone"

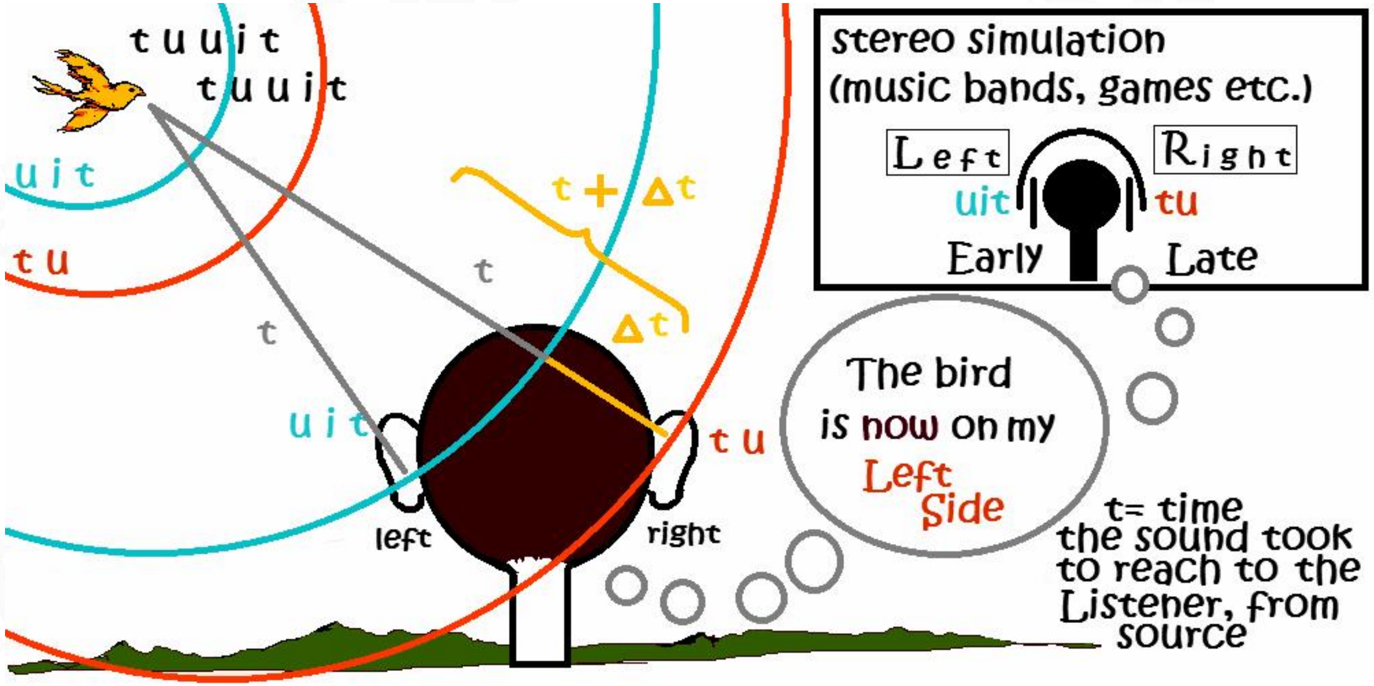
- position
- velocity
- orientation



Spatialization

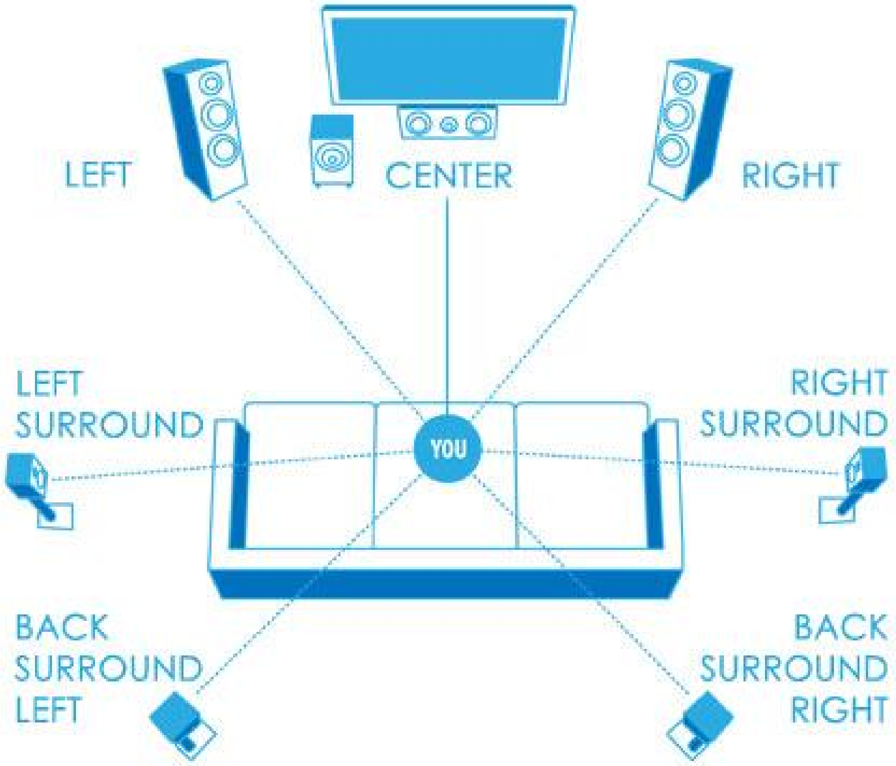
The techniques used to orient the sound relative to a listener

- Panning
- Soundfield
- Binaural Audio



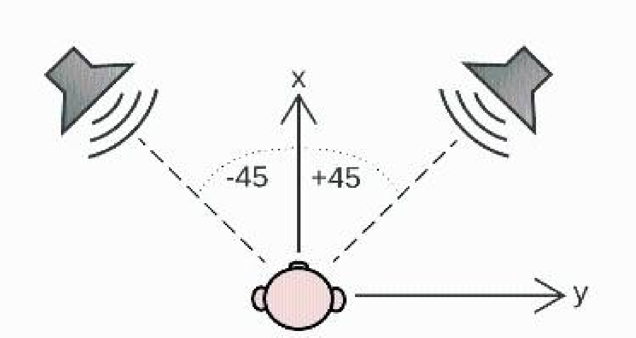
Panning - Channel

Distribution of an audio signal into a new stereo or multi-channel sound field



Panning (1/5) - Linear Panning

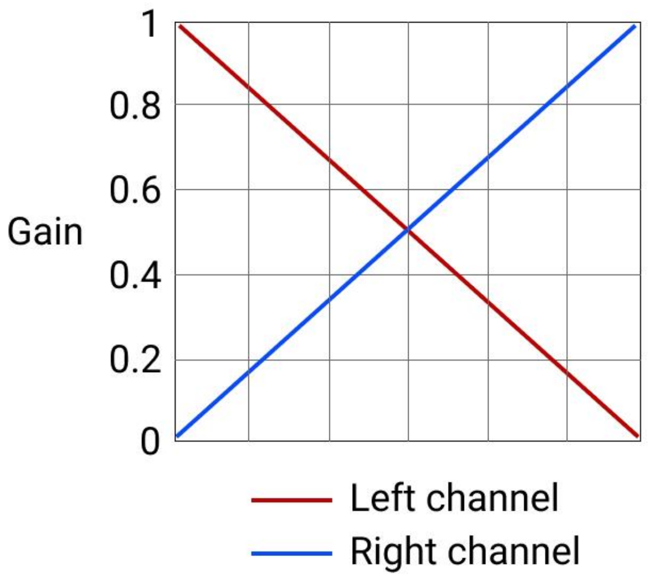
- **Main idea:** for a stereo signal with gain 1, the gains of the left and right channels should sum to 1



$$Gain_{left} = x$$

$$Gain_{right} = 1 - x$$

$$Gain_{left} + Gain_{right} = 1$$



Panning (2/5) - Linear Panning

- Human perception of loudness is actually proportional to the power of a sound wave
- Power is equal to the square of the signal's amplitude

$$Power_{right} = Gain_{right}^2 = (1-x)^2$$

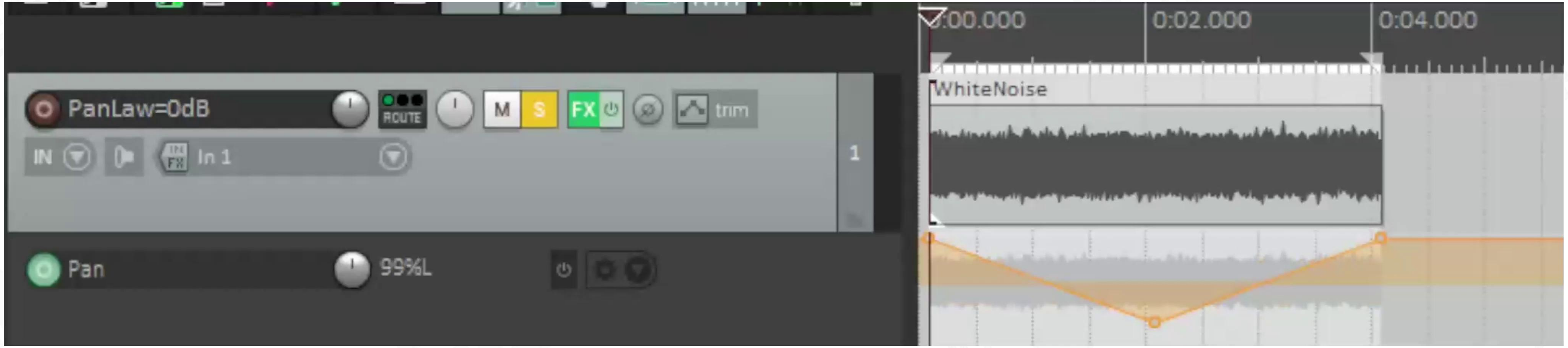
$$Power_{left} = Gain_{left}^2 = x^2$$

$$Power_{total} = x^2 + (1-x)^2$$

Panning (3/5) - Linear Panning

- The power will drop when the sound is panned in the middle(x = 0.5)

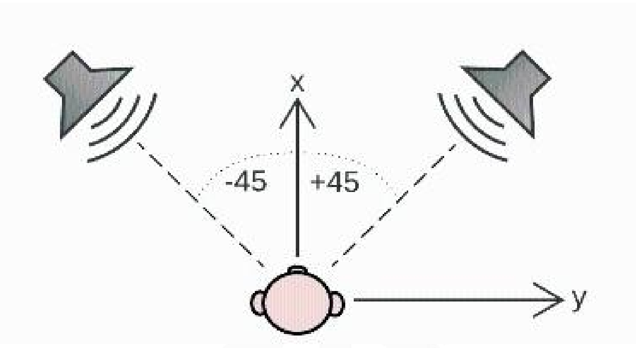
$$Power_{total} = (0.5)^2 + (1 - 0.5)^2 = 0.5$$



Panning (4/5) - Equal Power Panning

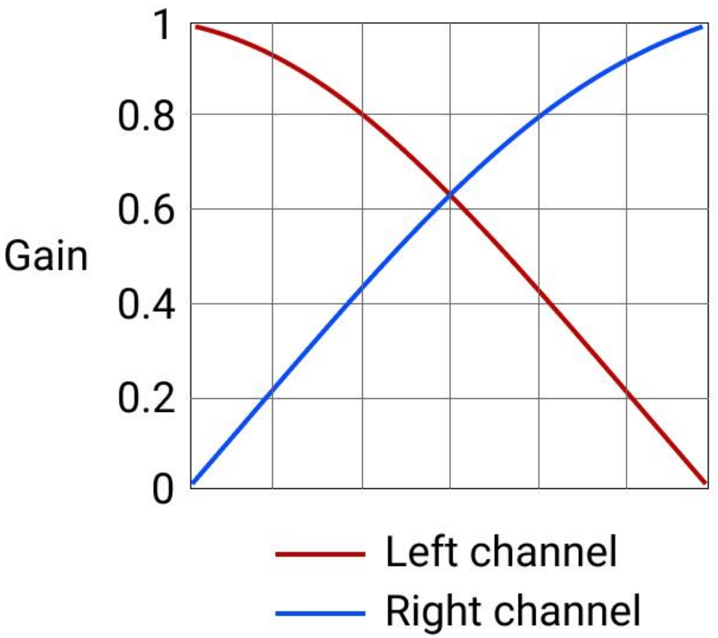
- Retain constant loundness by holding the power constriant during the pan, instead of holding the amplitude constant

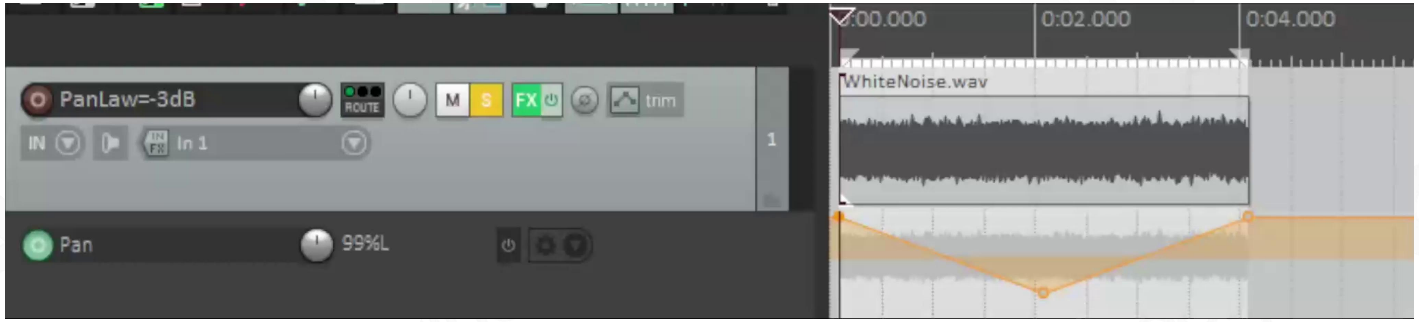
$$Power_{total} = Gain_{left}^2 + Gain_{right}^2 = 1.0$$



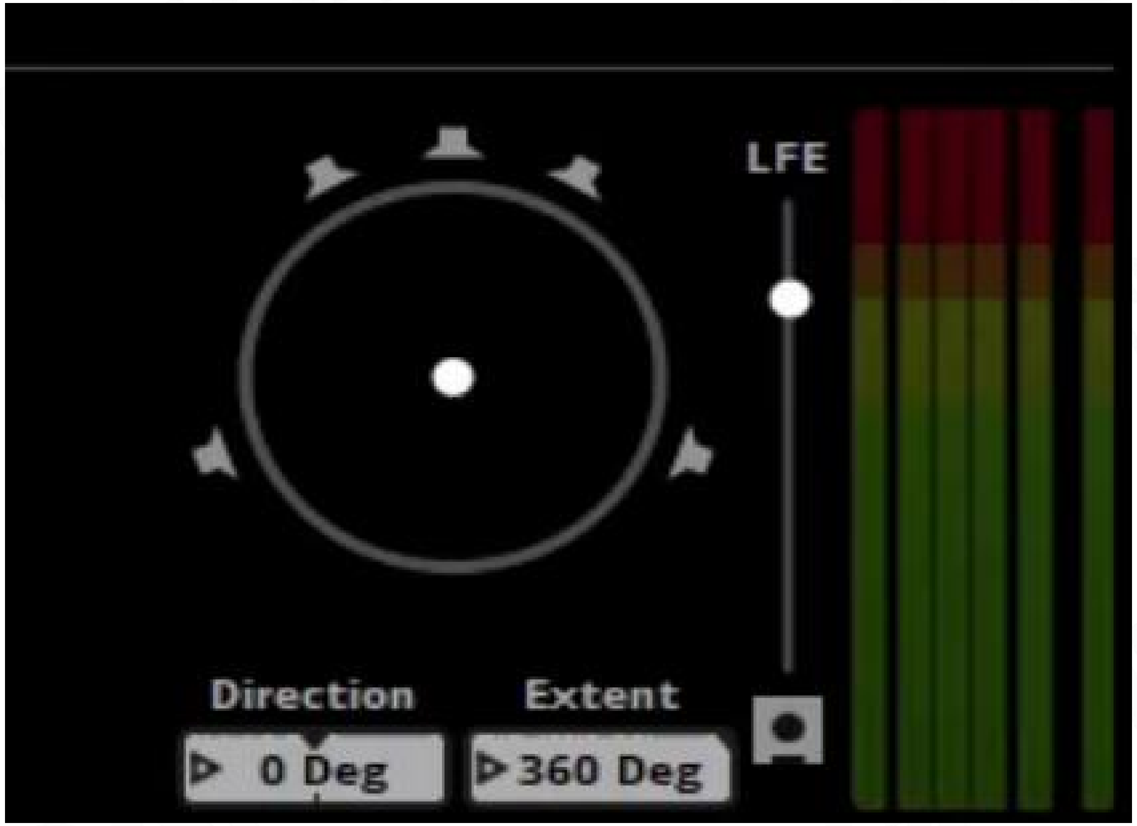
- Thera are several possible solutions to this equation, one is a sine/cosine equation

$$Gain_{left}^2 = sin^2(x) \qquad Gain_{right}^2 = cos^2(x)$$





Panning (5/5)

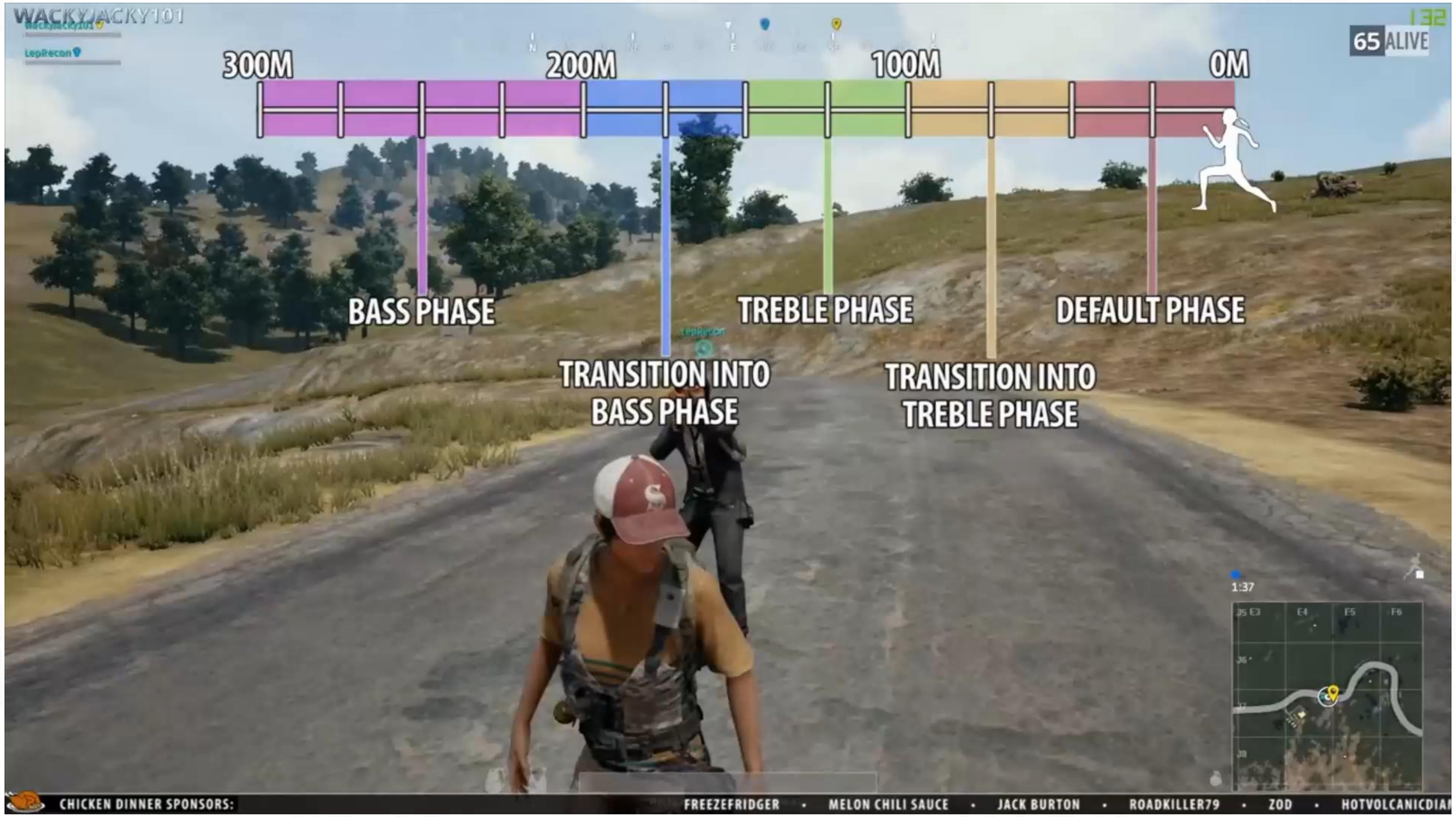


Attenuation

Volume will attenuate as the listener moves away from it

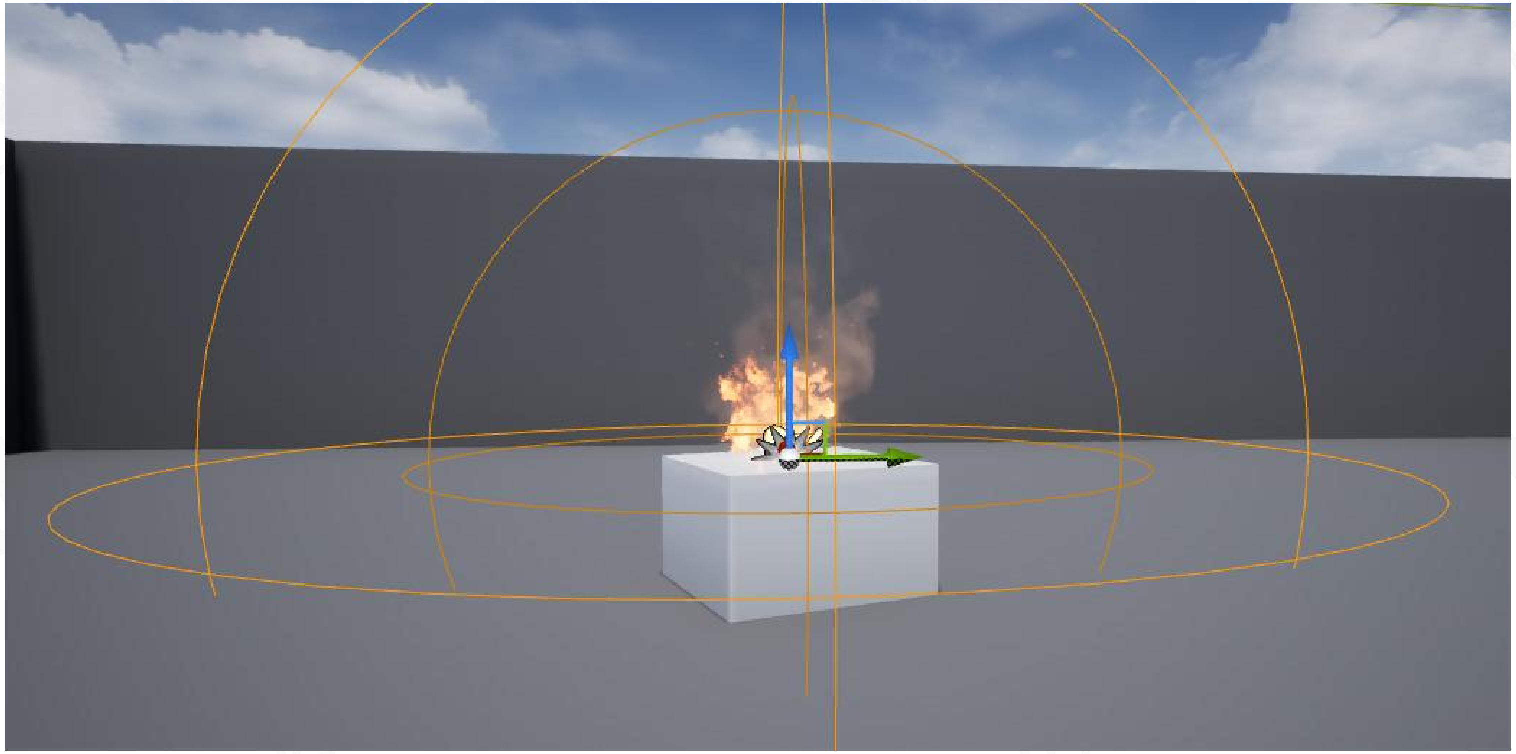
In real world, the sound pressure (p) of a spherical sound wave decreases as $1/r$ from the centre of the sphere:

$$p(r) \propto \frac{1}{r}$$



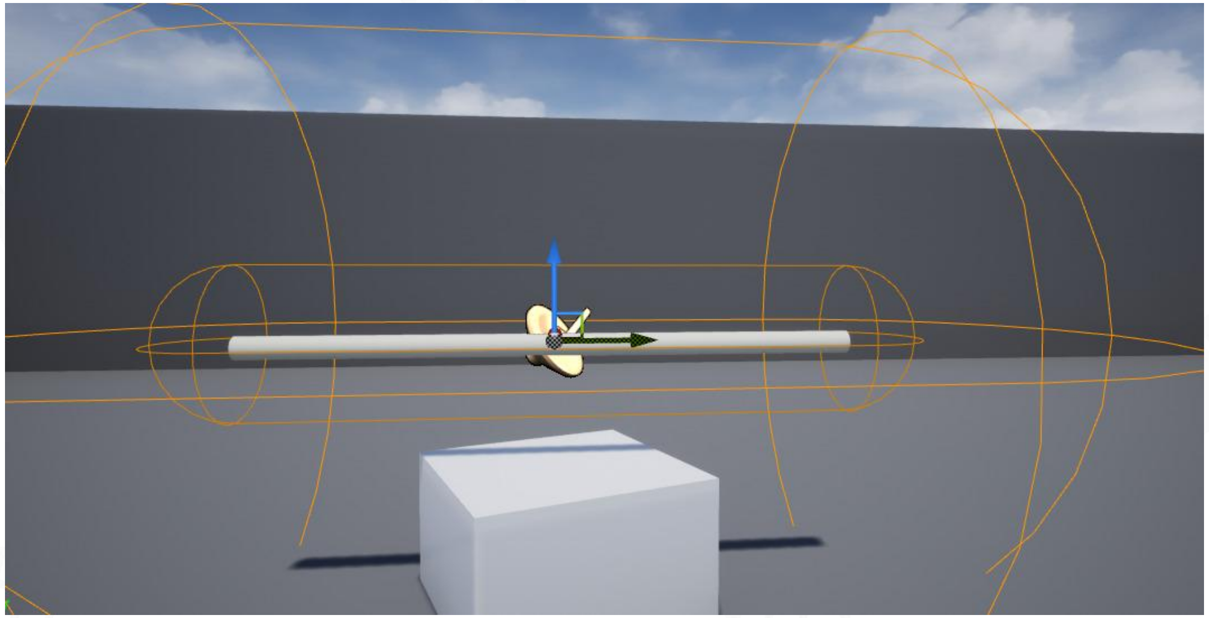
Attenuation Shape - Sphere

Useful for most spot sounds as it models how sound propagates in the real world



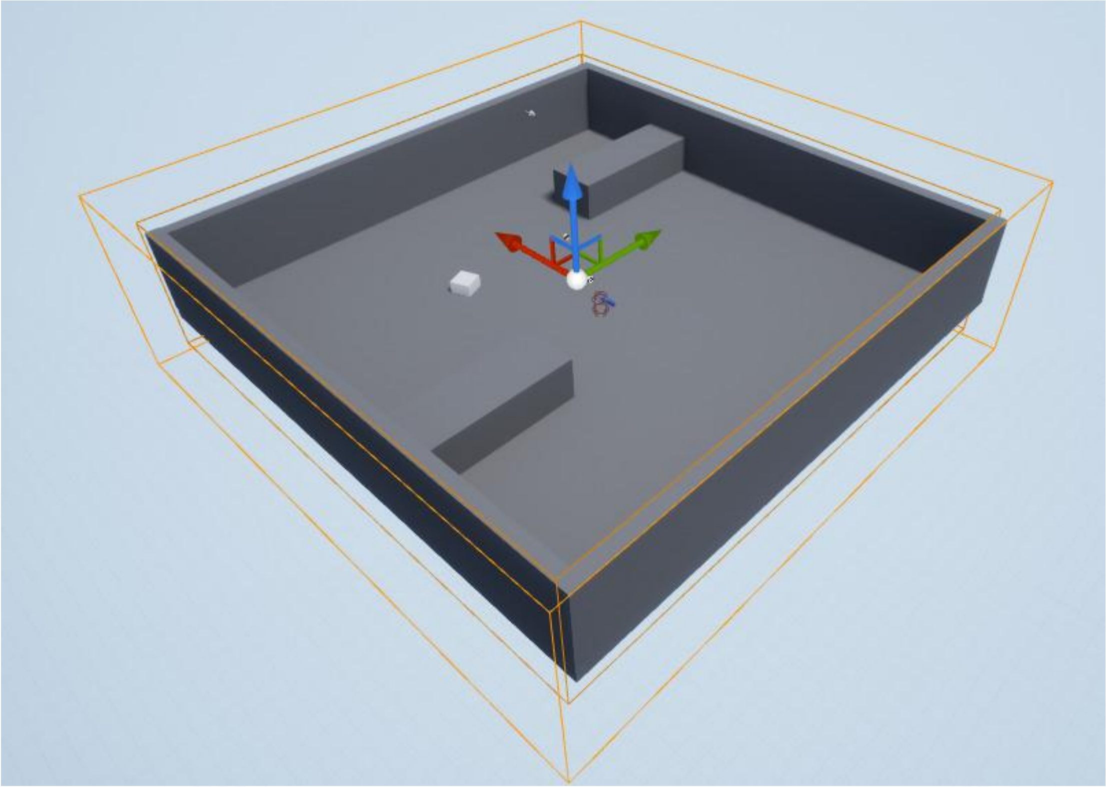
Attenuation Shape - Capsule

Useful for things like water pipes, where the sound doesn't want to appear to come from a single, specific point in space — the sound of gurgling water would follow the length of the pipe



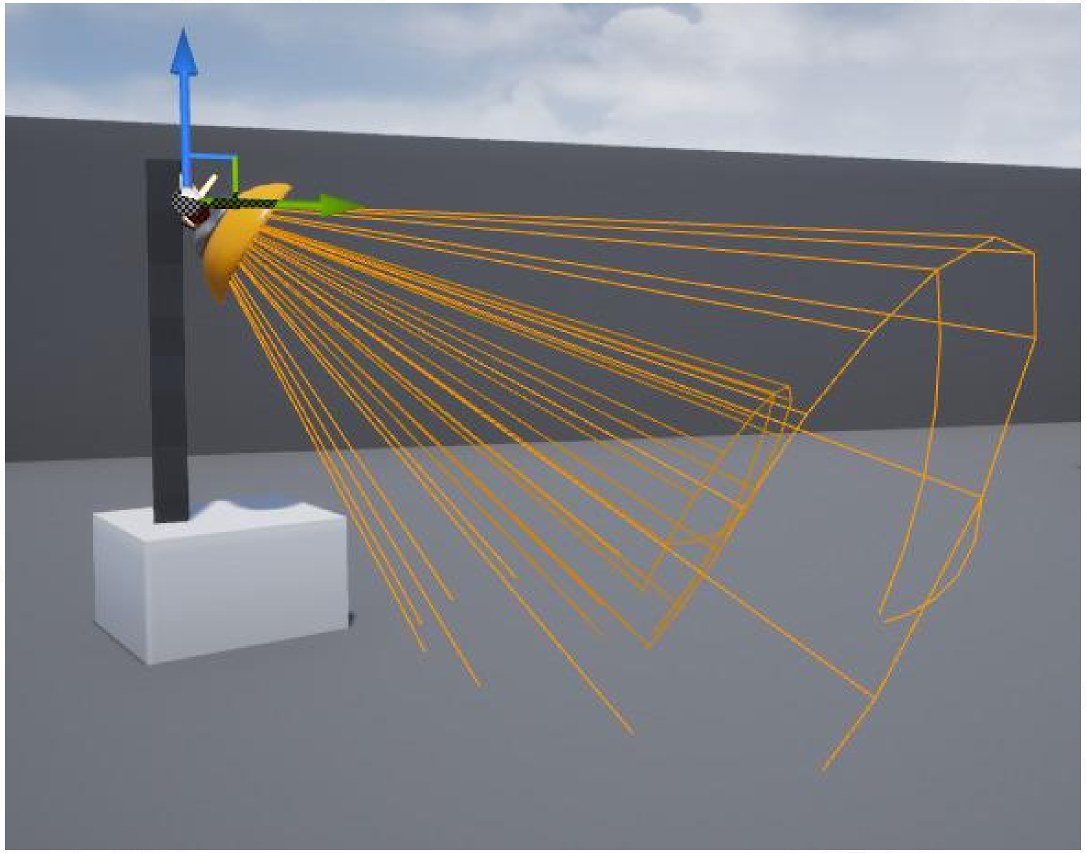
Attenuation Shape - Box

useful for things like room tones/ambiance as you can define the shape of the box to match that of the room

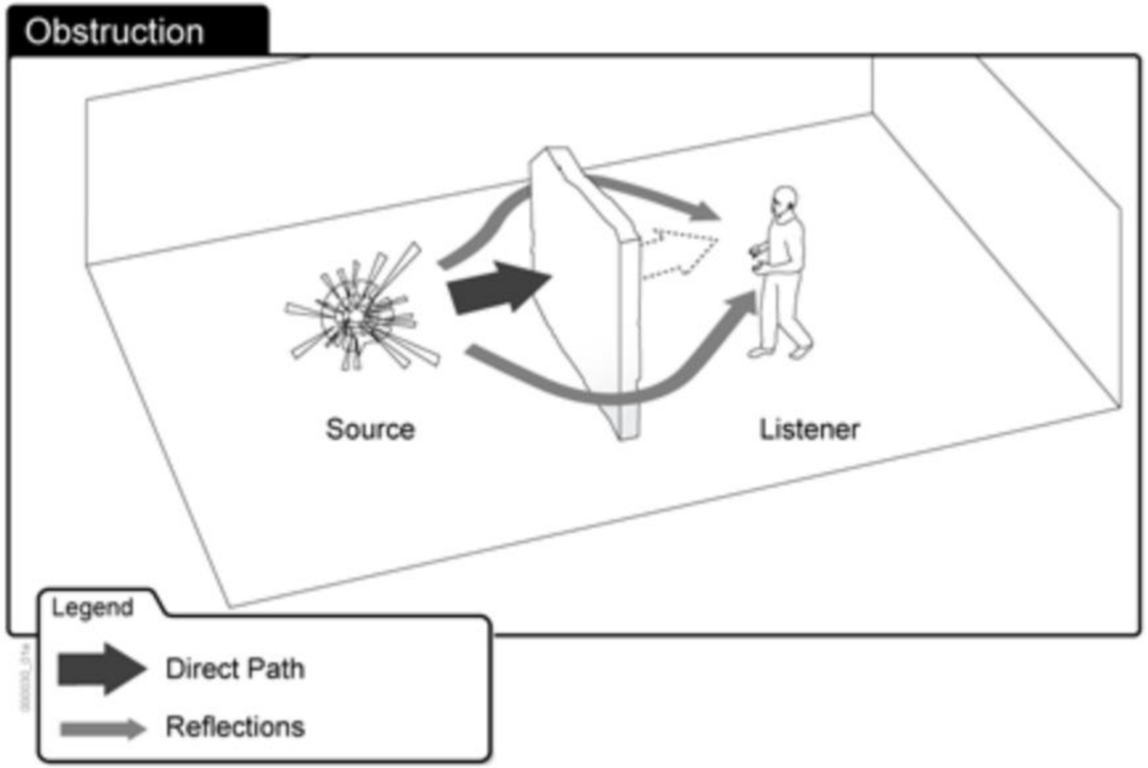


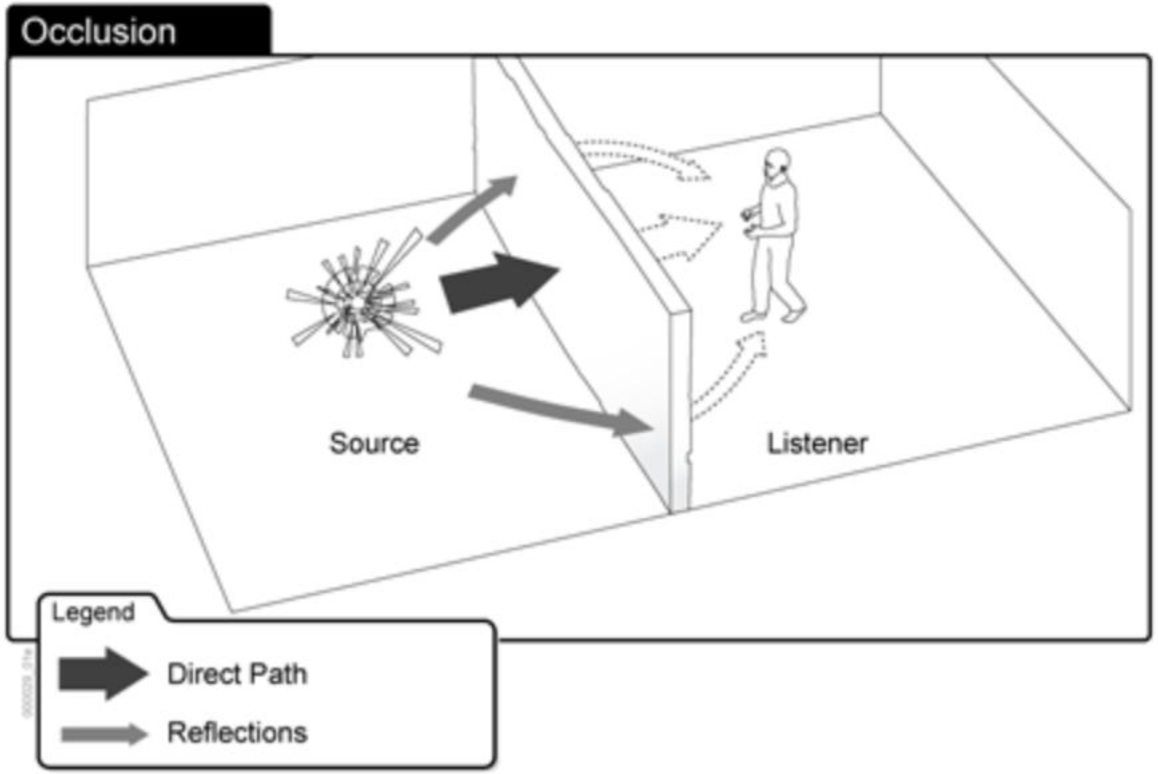
Attenuation Shape - Cone

Useful in situations when you want a directional attenuation pattern — for example, public address speakers



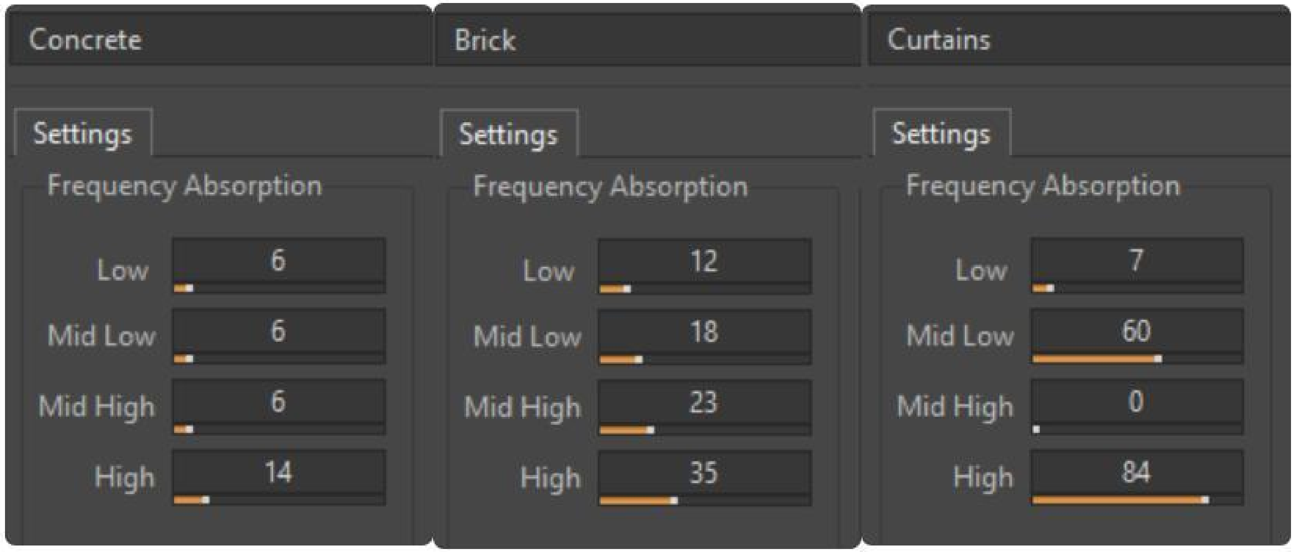
Obstruction and Occlusion（1/2）





Obstruction and Occlusion（2/2）

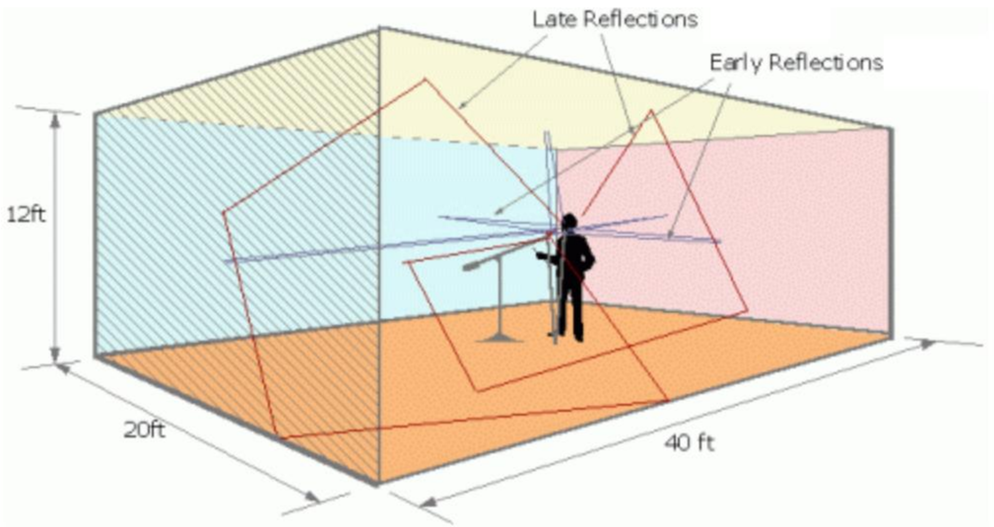
1. Cast a few divergent rays from listener to sound with different angle
2. query the material properties of the impacted surface to determine how much of the sound's energy it absorbs by the count of the blocked rays

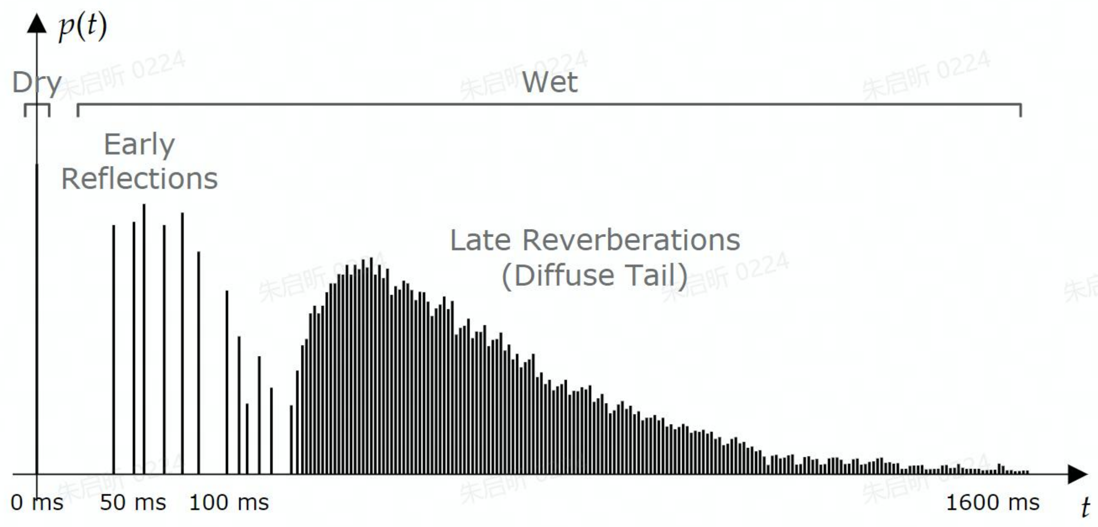


Reverb (1/3)

In any environment containing sound-reflective surfaces, a listener generally receives three kinds of sound waves from a sound source

- Direct (dry)
- Early reflections (echo)
- Late reverberations (tail)





Reverb (2/3)

Reverberation Time: Measure of how fast the sound dies away in a given room. The size of the room and the choice of materials determine the reverberation time

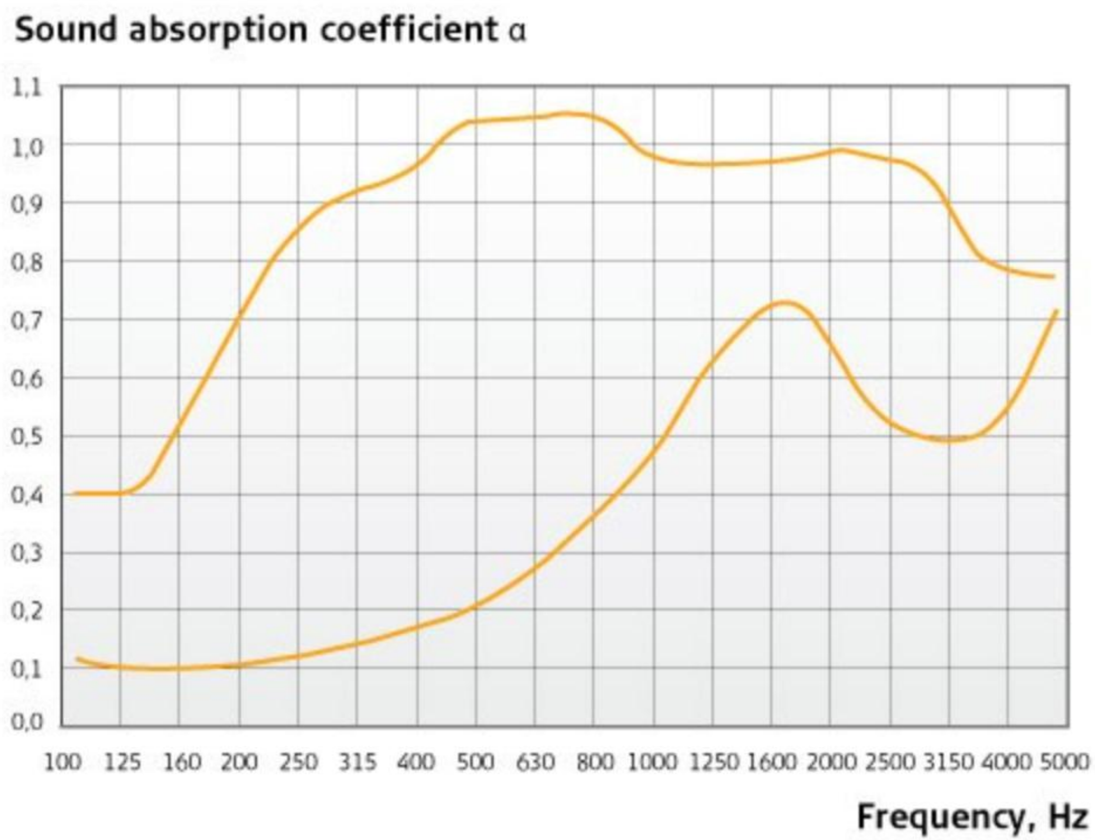
Absorption: Absorption coefficient of a material and material count determine the absorption

$$A = S \times \alpha (m^2 sab)$$

$$T = 0.16 \times \frac{V}{A} (s)$$

α: absorption coefficient
S: square
A: equivalent absorption area
V: volume in the room
T: reverberation time
0.16： proportionality factor, the time (in seconds) it takes for the initial sound pressure level to be reduced by 60 dB

Reverb (3/3)



Materials/Locations	Sound Absorption Coefficients over 1/1 Octave Bands					
	125 Hz	250 Hz	500 Hz	1 kHz	2 kHz	4 kHz
Ornamented stone piers, arches columns, and column heads	0.05	0.05	0.06	0.09	0.11	0.11
Current carpet	0.02	0.12	0.25	0.38	0.55	0.58
Carpet with prayers	0.18	0.22	0.41	0.58	0.69	0.72
Historical plasters tested in 30% humidity	0.10	0.17	0.23	0.29	0.32	0.32
Current plasters on brick	0.13	0.09	0.07	0.05	0.03	0.04
Large pane of glass	0.18	0.06	0.04	0.03	0.02	0.02
Marble slabs	0.01	0.01	0.01	0.01	0.02	0.02
Wooden doors and furniture	0.10	0.07	0.05	0.04	0.04	0.04
Gypsum moldings for muqarnas and similar decorations	0.29	0.10	0.05	0.04	0.07	0.09
Copper decorative elements	0.12	0.08	0.02	0.01	0.01	0.01

Sound absorption coefficient and scattering data of applied materials in acoustical simulations

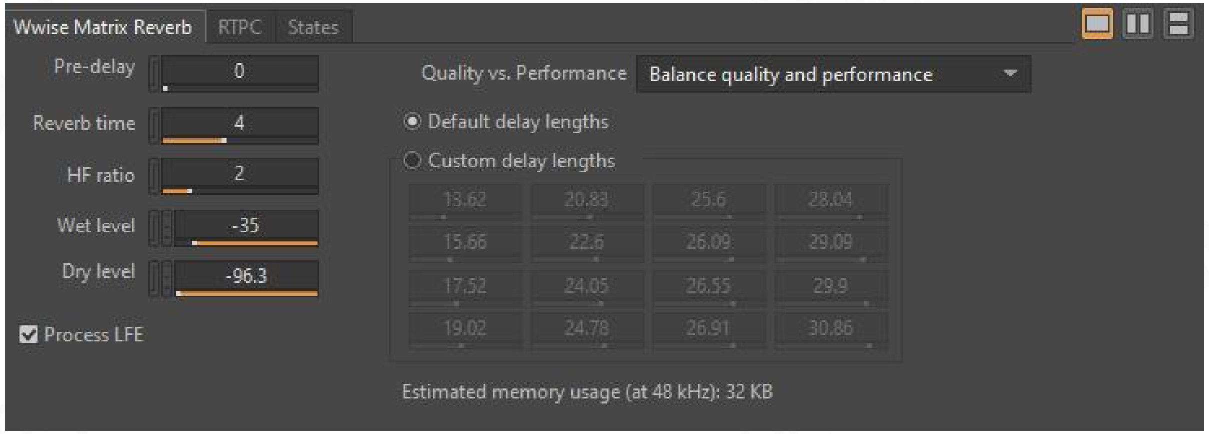
Reverb in Action - Reverb Effect Control from Acoustic Parameters

Pre-delay(seconds): The delay that occurs before the signal enters the reverberation unit. A longer pre-delay time can be used to simulate larger rooms where the first echoes take longer to be heard.

HF ratio: A rolloff factor to control the reverberation time for high relative to low frequencies.

Wet level: Gain factor applied to reverberated sound.

Dry level: Gain factor applied to direct path sound.



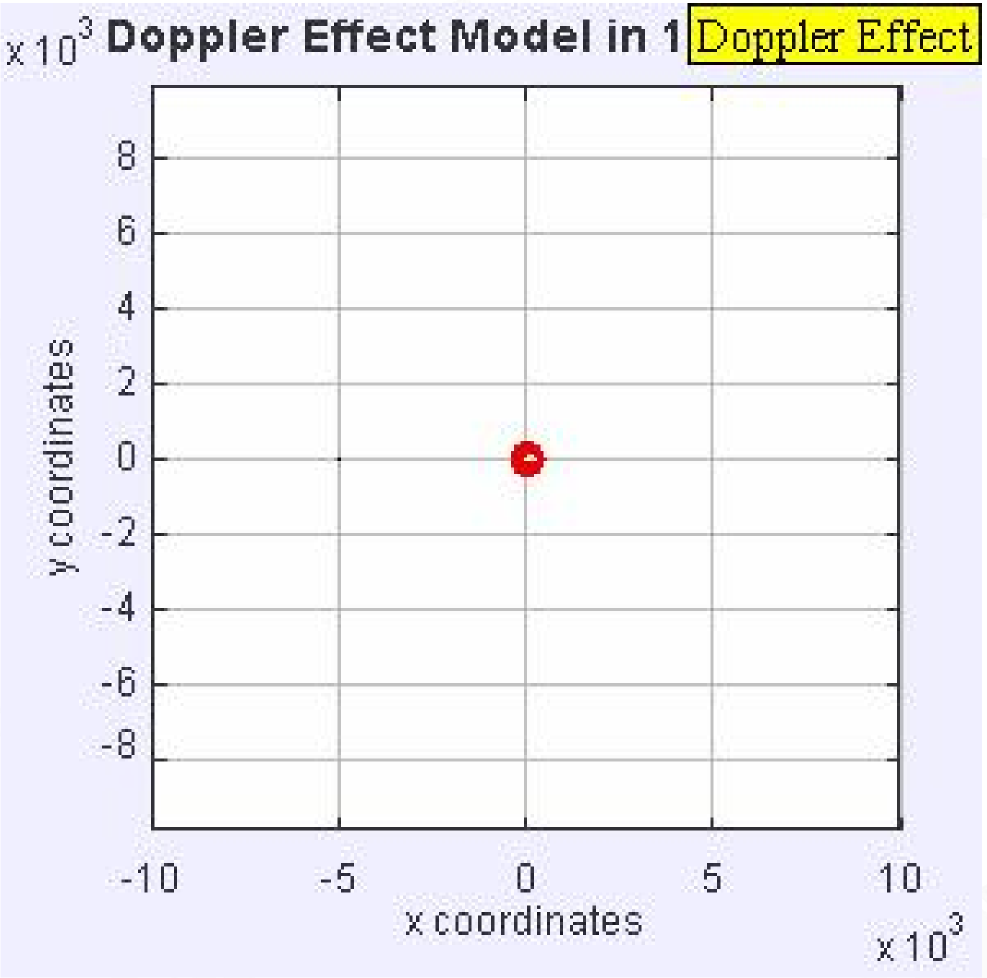
Sound in Motion: The Doppler Effect

The change in frequency of a wave in relation to an observer who is moving relative to the wave source

Sound in Motion: The Doppler Effect

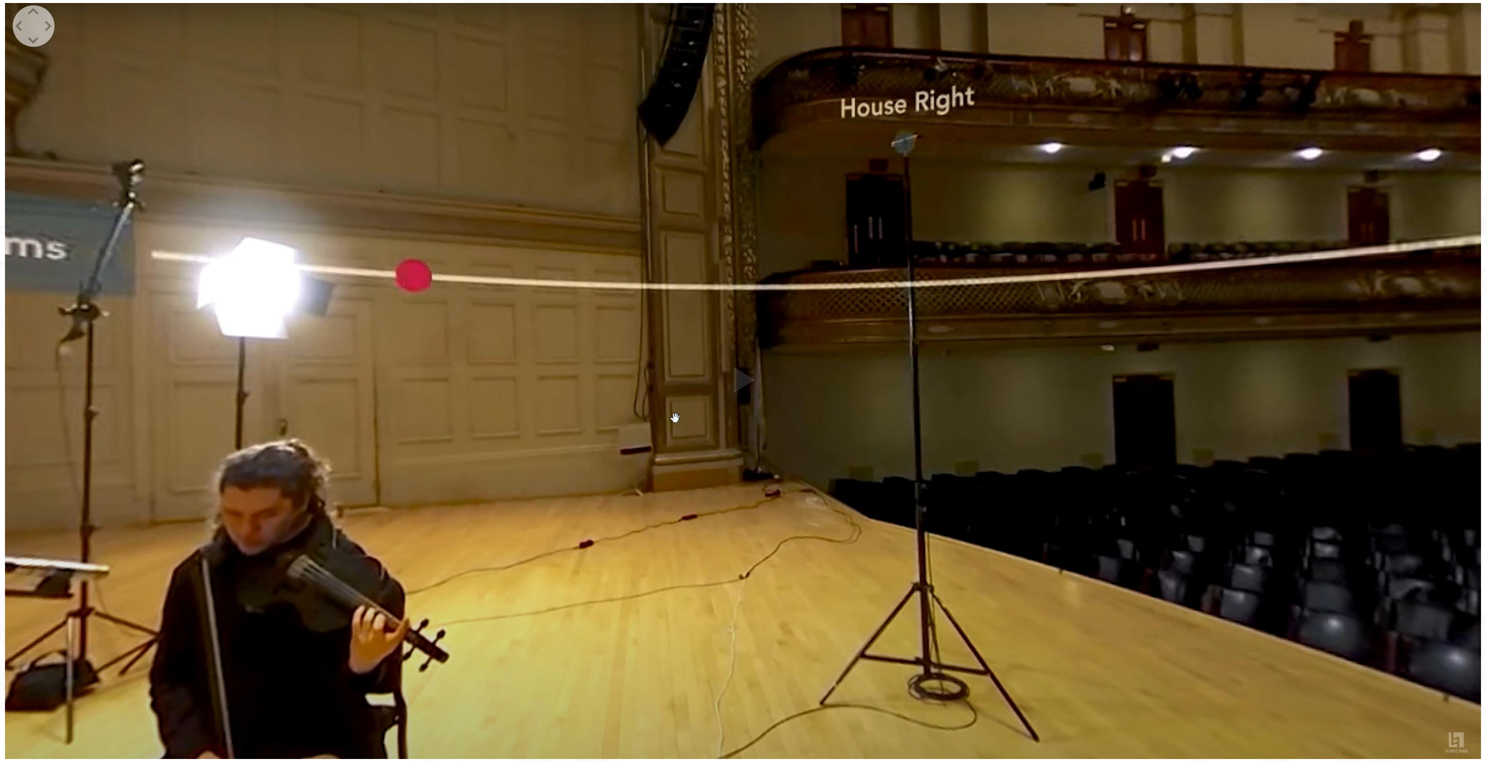
$$f' = \left(\frac{v + v_0}{v + v_s}\right) f$$

- f: original frequency
- f': Doppler-shifted (observed) frequency at the listener
- v: the speed of sound in air
- v0: the speed of the listener
- vs: the speed of the sound source



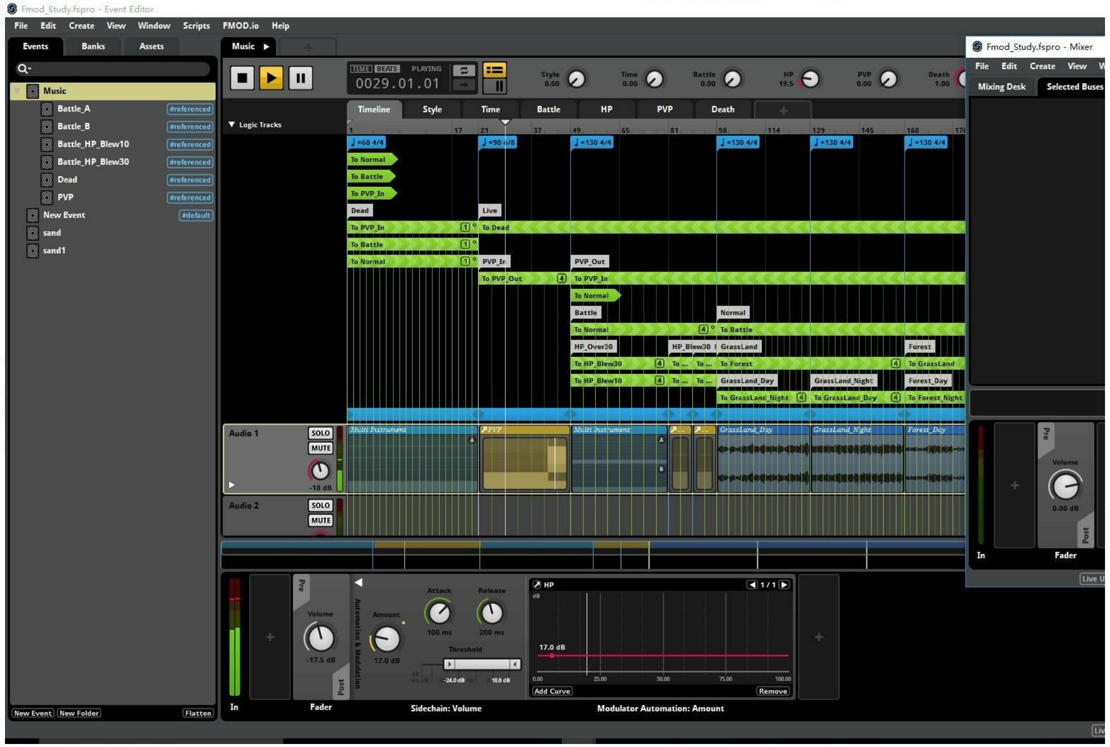
Spatialization - Soundfield

- Full-sphere surround sound
- Also known as ambisonics
- Used in 360 videos and VR

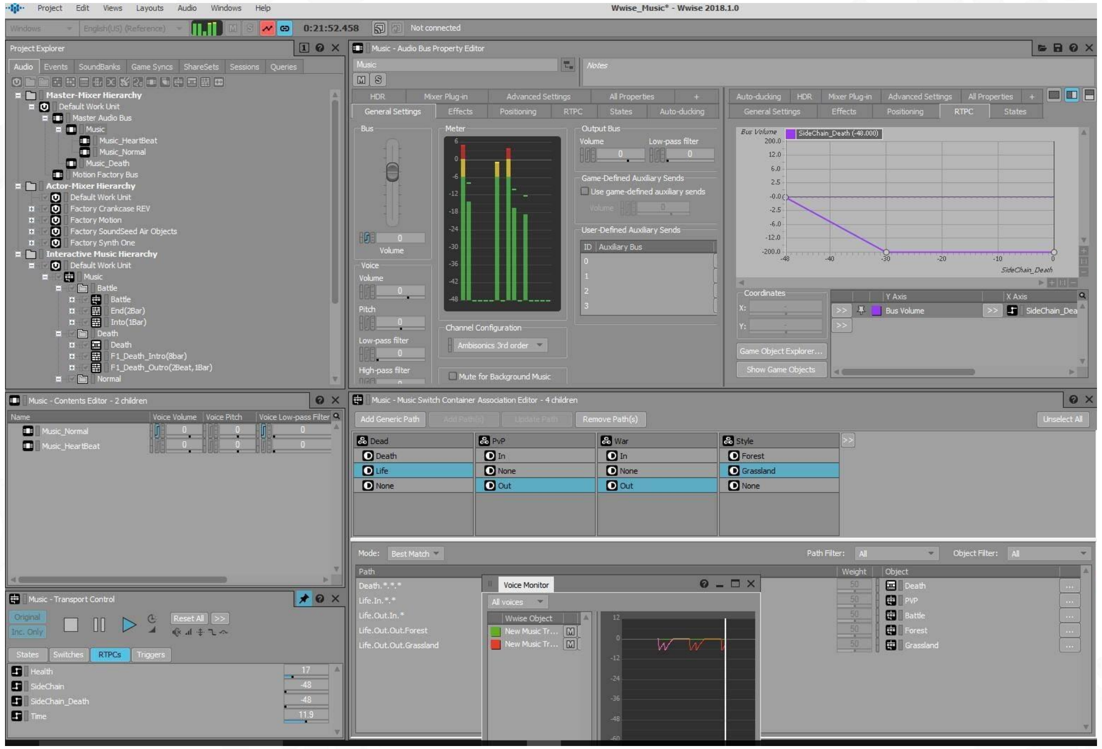


Common Middlewares

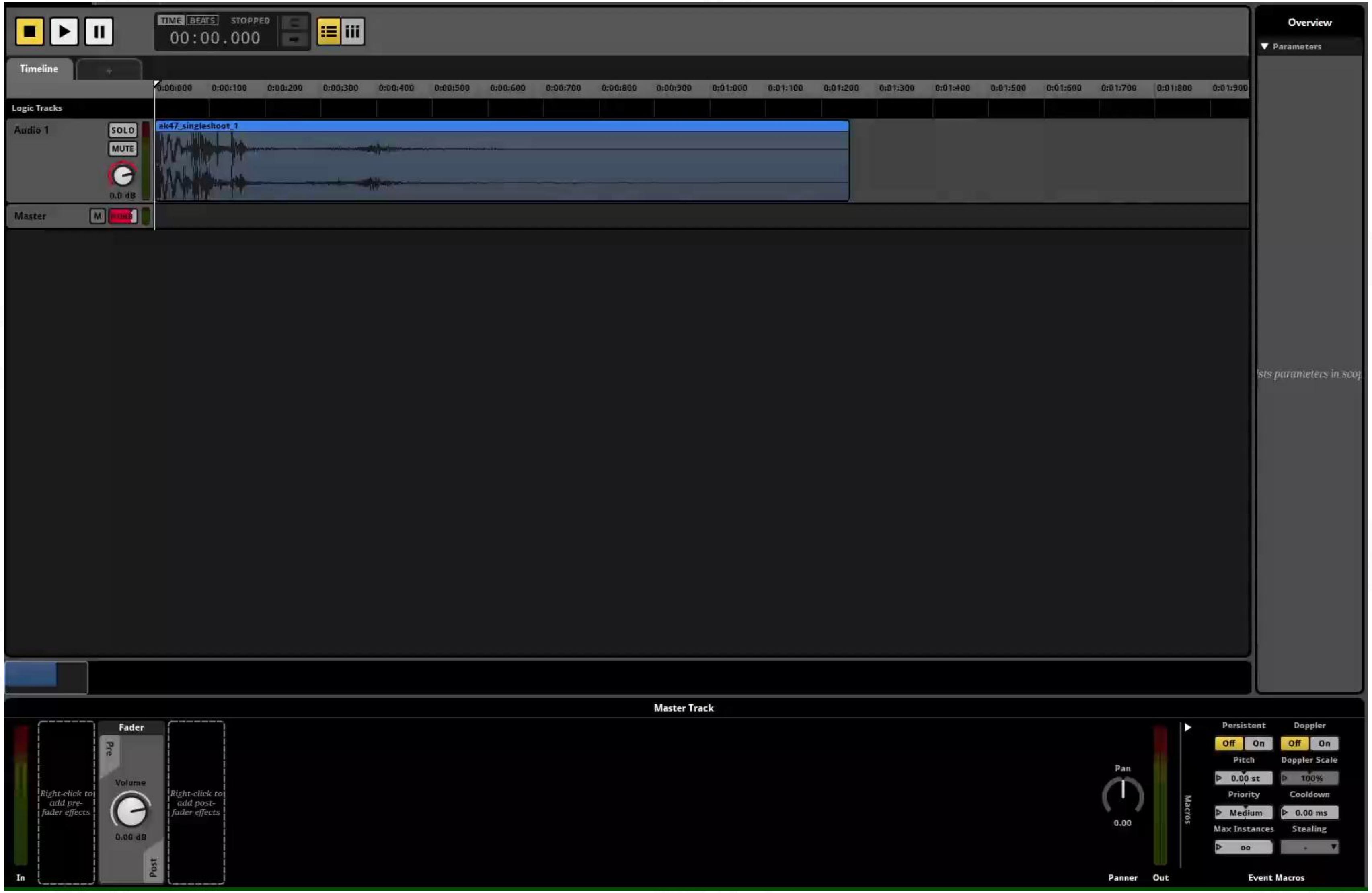
fmod



Wwise

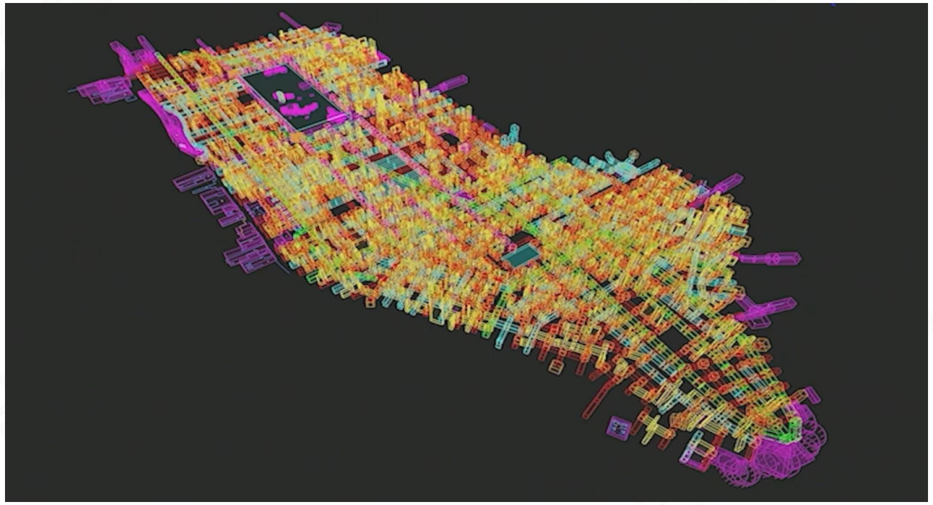


How does Audio Middleware Work?



Modeling Audio World

- Geometry and properties of the surfaces and object
- Acoustic properties of the listening spaces



Particle System

Programmable VFX with Unreal Engine's Niagara – GDC 2018:
https://www.unrealengine.com/en-US/events/gdc2018/programmable-vfx-with-unreal-engine-s-niagara

The Destiny Particle Architecture – SIGGRAPH 2017:
https://advances.realtimerendering.com/s2017/Destiny_Particle_Architecture_Siggraph_Advances_2017.pptx

Frostbite GPU Emitter Graph System – GDC 2018:
http://www.gdcvault.com/play/1025132/Frostbite-GPU-Emitter-Graph

The inFAMOUS: Second Son Particle System Architecture – GDC 2014:
https://www.gdcvault.com/play/1020367/The-inFAMOUS-Second-Son-Particle

Compute-Based GPU Particle Systems – GDC 2014:
https://www.gdcvault.com/play/1020002/Advanced-Visual-Effects-with-DirectX

The Visual Effects of inFAMOUS: Second Son – GDC 2014:
https://www.gdcvault.com/play/1020158/The-Visual-Effects-of-inFAMOUS

Mergesort - Modern GPU:
https://moderngpu.github.io/mergesort.html

A Faster Radix Sort Implementation – Nvidia:
https://developer.download.nvidia.cn/video/gputechconf/gtc/2020/presentations/s21572-a-faster-radix-sort-implementation.pdf

Audio System

Designing the Bustling Soundscape of New York City in ‘Marvel’s Spider-Man’ – GDC 2019:
https://www.gdcvault.com/play/1026515/Designing-the-Bustling-Soundscape-of

An Interactive Sound Dystopia: Real-Time Audio Processing in ‘NieR:Automata’– GDC 2018:
http://www.gdcvault.com/play/1025132/Frostbite-GPU-Emitter-Graph

Game Audio Programming in C++ – CppCon:
https://www.youtube.com/watch?v=M8Bd7uHH4Yg

Spatialization Overview :
https://docs.unrealengine.com/5.0/en-US/spatialization-overview-in-unreal-engine/’

Sound Attenuation:
https://docs.unrealengine.com/5.0/en-US/sound-attenuation-in-unreal-engine/

A Wwise Approach to Spatial Audio – Part1 – Distance Modeling and Early Reflections:
https://blog.audiokinetic.com/zh/a-wwise-approach-to-spatial-audio-part-1/

A Wwise Approach to Spatial Audio – Part1 – Diffraction:
https://blog.audiokinetic.com/zh/a-wwise-approach-to-spatial-audio-part-2-diffraction/

A Wwise Approach to Spatial Audio – Part1 – Beyond Early Reflections:
https://blog.audiokinetic.com/zh/a-wwise-approach-to-spatial-audio-part-3-beyond-early-reflections/

Lecture 12 Contributor

- 坤
- Uchihaxin
- 少年
- 嘉衡
- 小老弟
- 建辉
- 馨月
- 爵爷
- Jason
- 砚书
- BOOK
- MANDY
- 乐酱
- 灰灰
- 金大壮
- Leon
- 梨叔
- Shine
- 浩洋
- Judy
- 乐酱
- QIUU
- C佬
- 阿乐
- 靓仔
- CC
- 大喷
- 大金

Q&A

Enjoy ;) Coding

Course Wechat

Follow us for further information

Please note that all videos and images and other media are cited from the Internet for demonstration only.