# COMP30019 Graphics and Interaction Particle Systems

Adrian Pearce

Department of Computer Science and Software Engineering
University of Melbourne

The University of Melbourne

# Lecture outline

Aim: understand particle systems
Reading:

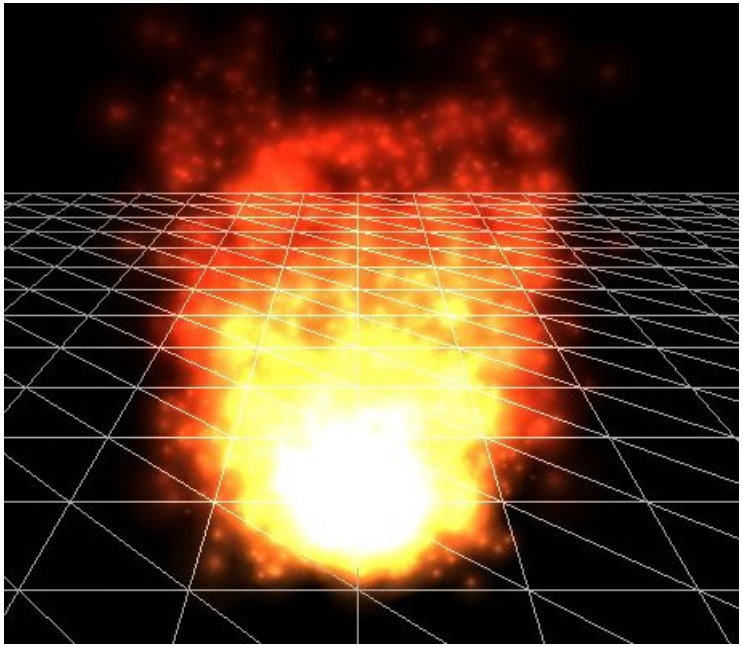
- Particle Systems (Wikipedia) `https://en.wikipedia.org/wiki/Particle_system`
- Unity3D Particle system tutorials: `https://www.youtube.com/watch?v=2aobHHSQgjY`

# Particle Systems

Particle systems can be used to mode amporphous objects, such as

- Fire,
- Smoke,
- Water,
- Clouds,
- etc.

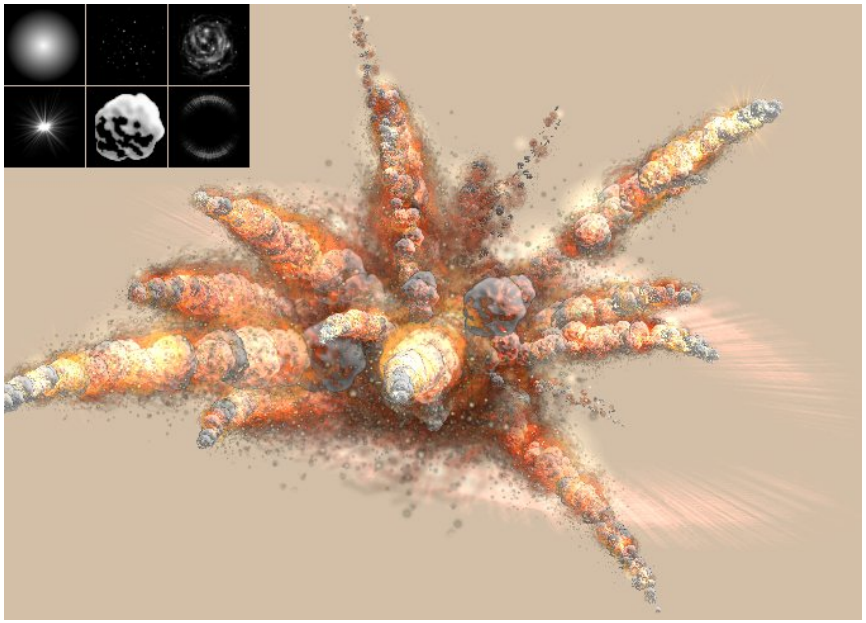
# Particle Systems



A particle system used to simulate a fire, created in *3dengfx*—created by creating particles and introducing secondary light sources (at point of fire)

Example (from Koldora): `https://www.youtube.com/watch?v=mMHcnyZMp-c`

# Particle Systems



A particle system used to simulate a bomb explosion, created in *particleIllusion*.

# Particle creation

Discrete set of small particles, can have

- Mass (for dynamic model)
- Position
- Velocity
- Shape
- Lifetime

Examples of different parameters by : `https://www.youtube.com/watch?v=heW3vn1hP2E`

# Particle creation

- Particles can be created using a *normal* probability distribution.
- They can be given an *initial velocity* and a *lifetime*
- Depending on the simulation their movement can be determined by a dynamic model.

# Particle creation: water (example)

- New particles created at each frame (number created can be normally distributed)
- Each particle has initial downward velocity (again, normally distributed)—*here, each successive frame each particle is acted on by either force and 'wind'* Image from `http://www.iamag.co`

# Particle dynamics

Newtonian particles $f = ma$ (where $f$ and $a$ are vectors)

$$a = \frac{dv}{dt} = \frac{d^2x}{d^2t}$$

Given $f$ find the change in position $x$ so we need to *integrate*.

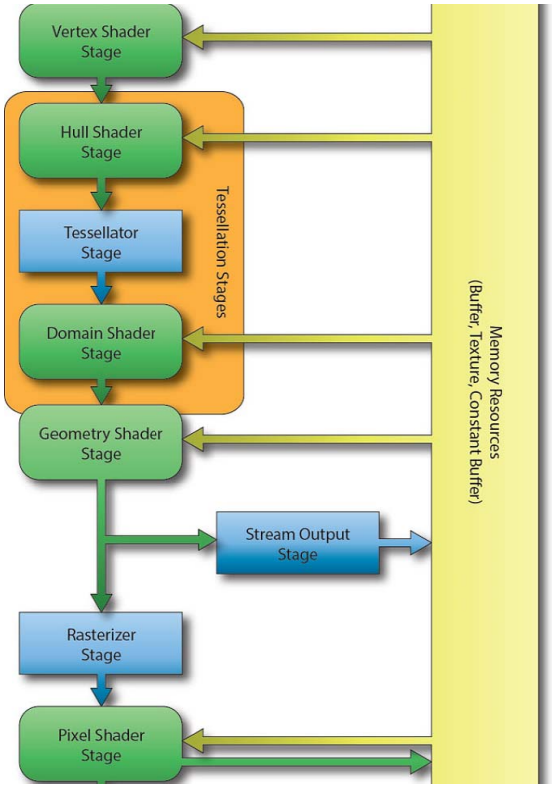
However using frame intervals we can use a simple approximation:

$$v_{t+1} = a_t \Delta t + v_t$$

$$x_{t+1} = v_t \Delta t + x_t$$

# Computation & Geometry Shaders

## DirectX 11 Pipeline



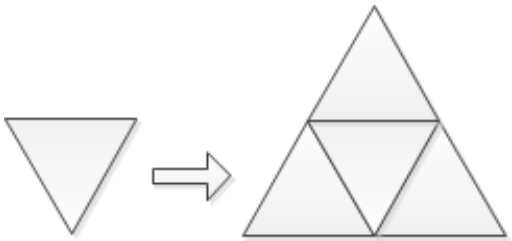

DirectX 11 Pipeline: `http://www.3dgep.com/introduction-to-directx-11/`

# Geometry Shaders

- The primary use of the geometry shader (introduced in DirectX10) is used for creating new primitives from existing ones
- It is most useful for use with *particles systems* and *sprite* rendering

# Primitives

- The geometry shader takes a whole *primitive* as input
- If primitive is set to ‘points’ then geometry shader only accepts a single vertex, while for ‘lines’ and ‘triangles’ it accepts 2 or 3 vertices respectively



**Primitive construction in geometry shader:** The geometry shader uses vertex data of the input primitive’s to create new vertices which form *new primitives*.

Newly created vertices are output as a vertex stream, continuing through the pipeline—Figure from `https://takinginitiative.wordpress.com/2011/01/12/directx10-tutorial-9-the-geometry-shader/`

# Types of Particle Systems

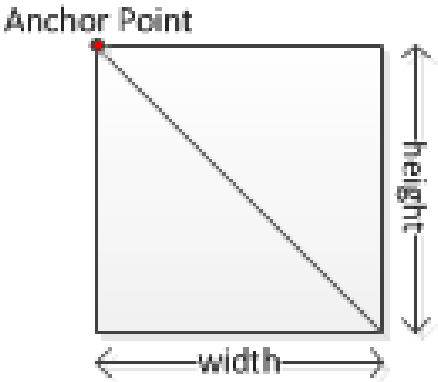
Particle simulations can be achived in two ways:

- Sprite (2D) particle systems
- Voxel (3D) particle systems

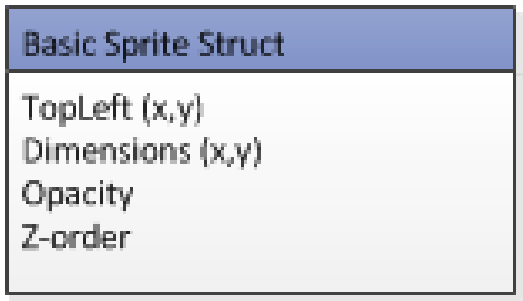
# Sprites

A *sprite* is a textured quad. Sprite dimensions and position are usually controlled by two sets of values,

- the anchor point position and
- the sprite dimensions.




(a)



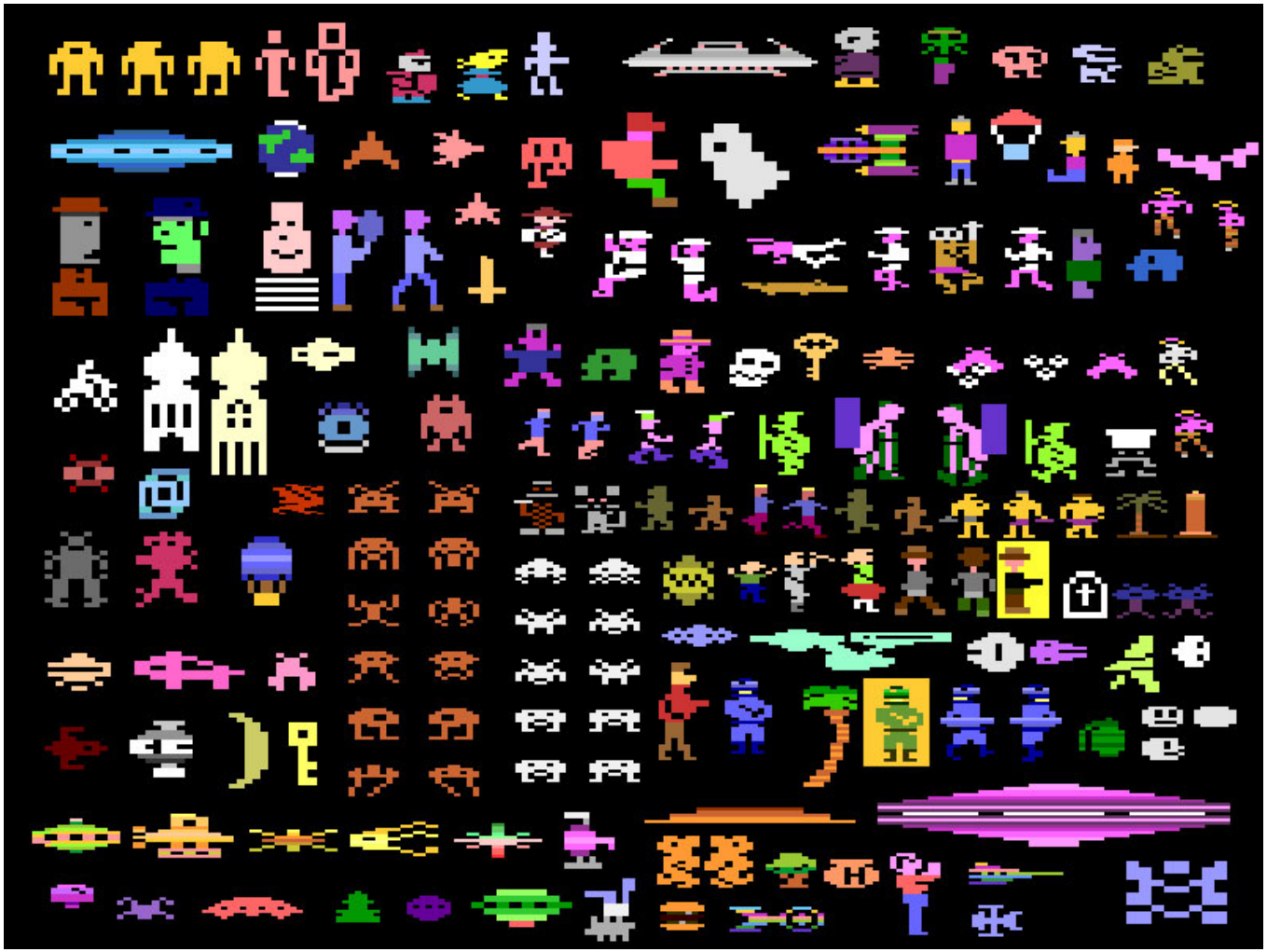

(b)

The *anchor point* (a) is used to position the sprite, while *dimension* (b) relates to width and height. `https://en.wikipedia.org/wiki/Sprite_(computer_graphics)`

# Sprites


Sprites from *Commodore 64* game (from Wikipedia)

# Sprites



Atari sprites from `www.vg-resource.com`

# Aside: AI Planning playing Atari

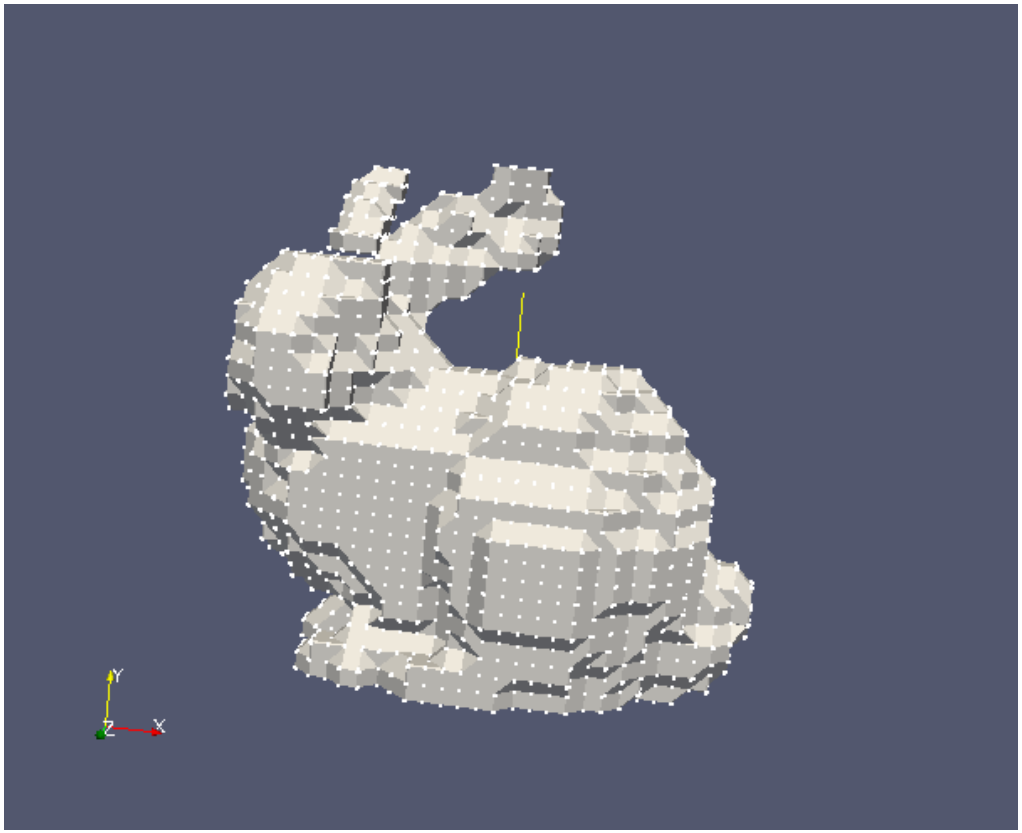
IW and 2BFS (AI Planning techniques) playing Atari:

`https://www.youtube.com/watch?v=P-603qPMkSg&list=PLXpQcXUQ_CwenUazUivhXyYvjuS6KQOI0`

Dr. Nir Lipovetzky—works in our AI and Autonomy Lab & teaches *COMP90054 AI Planning for Autonomy* in our Department

See `http://agentlab.cis.unimelb.edu.au/` for more details.

# Voxels




*Voxels* are essentially 3D pixels

`https://en.wikipedia.org/wiki/Voxel`

Image from `www.pointclouds.org`

# Voxels

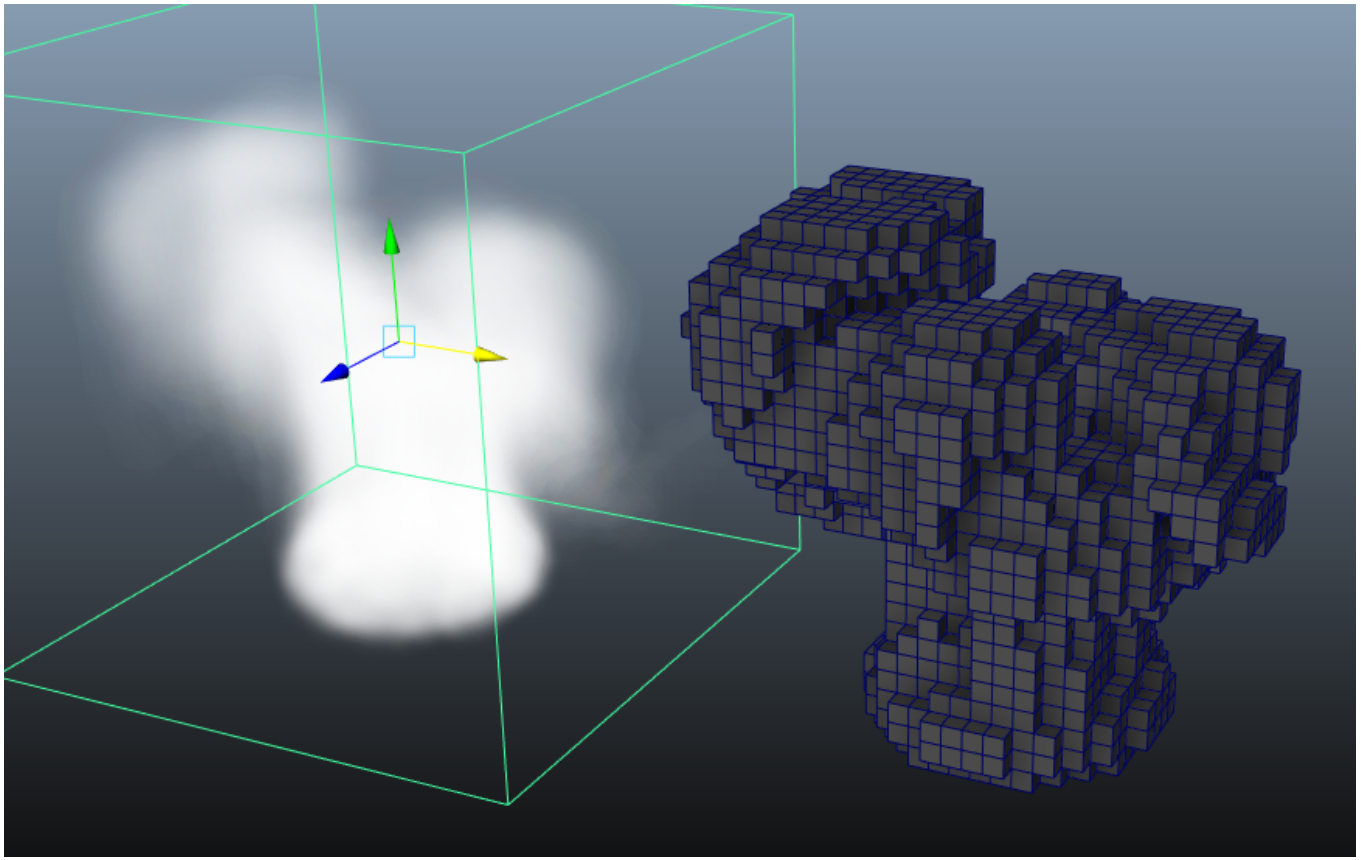
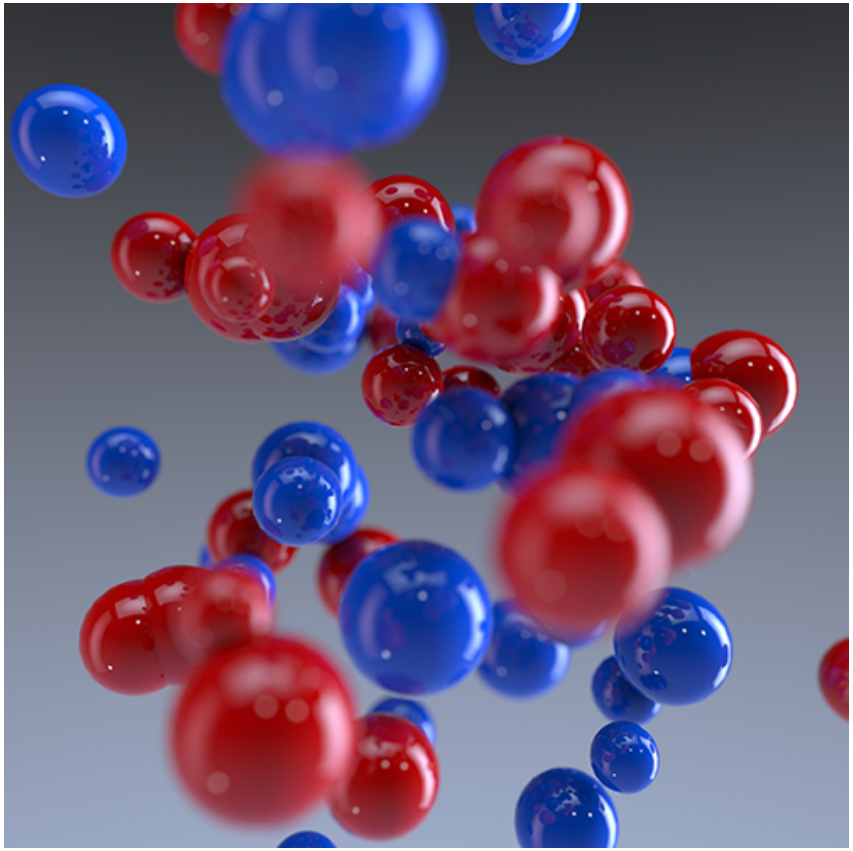


Image from `www.creativecrash.com`

# Voxels



Voxels are quite widely used in *cinema*—since online performance is less of an issue (can be computationally challenging!) Image from `solidangle.com`

# Voxels

10 Million particles

`https://www.youtube.com/watch?v=qaZIj6DUV8M`

Virtual Dancer: Hair, turblence and more:

`https://www.youtube.com/watch?v=miWXtX46QAY`

Source: Cinema4D

# Unity3D Particle system (since version 3.5)

View Unity3D Particle system tutorials:

`https://www.youtube.com/watch?v=2aobHHSQgjY`