Lesson 10 – Particle systems

PV227 – GPU Rendering

Jiří Chmelík, Jan Čejka
Fakulta informatiky Masarykovy univerzity

08. 12. 2015

Particle System

- general name of a large number of techniques that simulate natural phenomena such as smoke, dust, fireworks, rain, etc.
- composed of a large amount of small particles that move together in a way which is characteristic of each type of phenomenon
- we usually maintain the position as well as other attributes for each particle (velocity, color, etc) and perform the following steps once per frame:
  1. update the attributes of each particle – involves math calculations (ranging from very simple to very complex (depending on the complexity of the phenomenon))
  2. render the particles (as simple colored points or full blown texture mapped billboard quads).

Particle System . . .

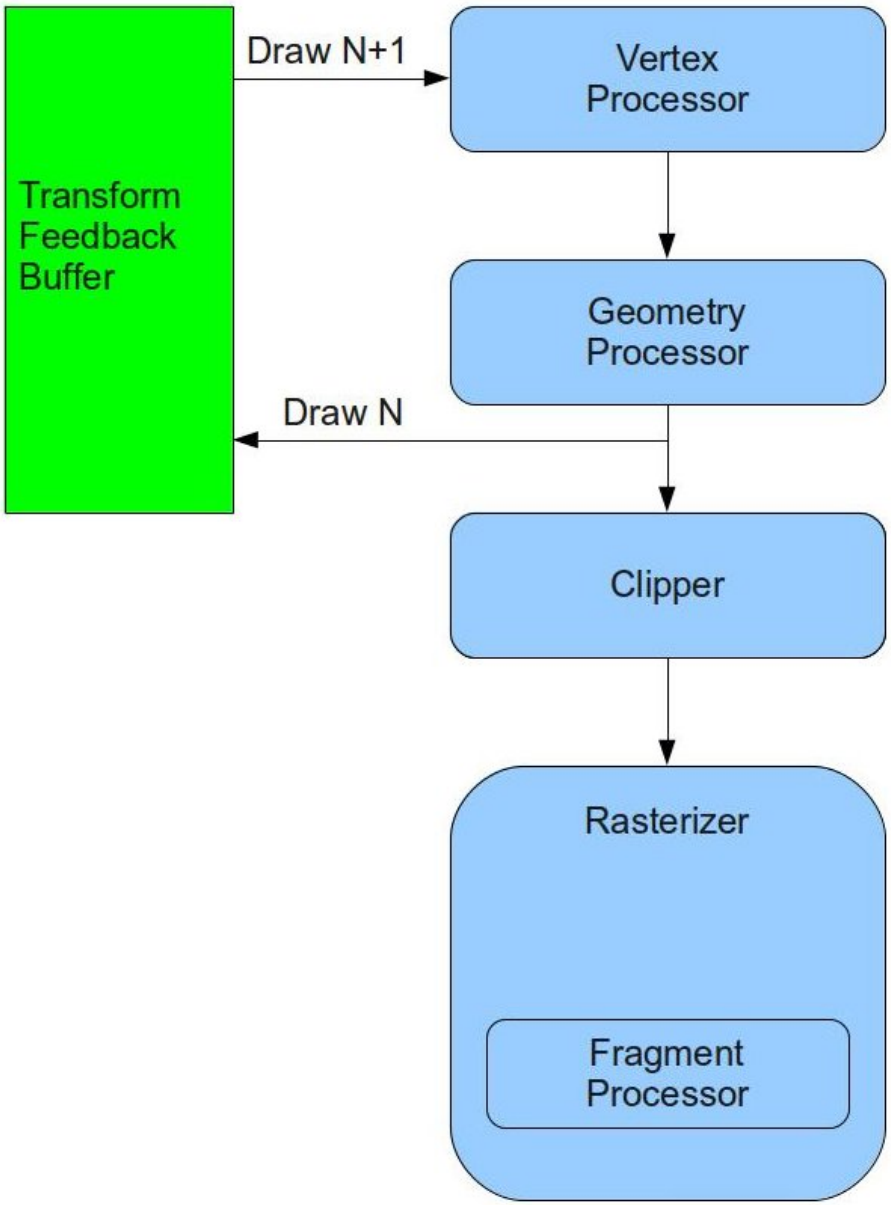
- Old way:
  - Update of particles on CPU, rendering on GPU.
  - pro – we can calculate whatever behaviour of particles we want
  - con – we need transfer data from CPU to GPU each frame $\rightarrow$ performance hit
- side-step way:
  - everything is solved on GPU
  - necessary data are stored in texture
  - pro – no need to transfer data
  - con – bulky solution, we are limited in what we can store in texture
- New way:
  - everything is solved on GPU
  - necessary data are stored in **Transform Feedback Buffer**
  - pro – more elegant solution

Transform Feedback Buffer

- special type of buffer where we can send transformed primitives from geometry shader (or vertex shader if there is no GS)
- plus – we can decide whether the primitives will also continue on their regular route to the rasterizer.
- The same buffer can be connected as a vertex buffer in the next draw and provide the vertices that were output in the previous draw as input into the next draw.

- We don't know number of primitives in buffer
- we don't care...

Transform Feedback Buffer scheme



Fixed Particles

- render the particle as a point,
- update their position based on time,
- optionally texture the point.

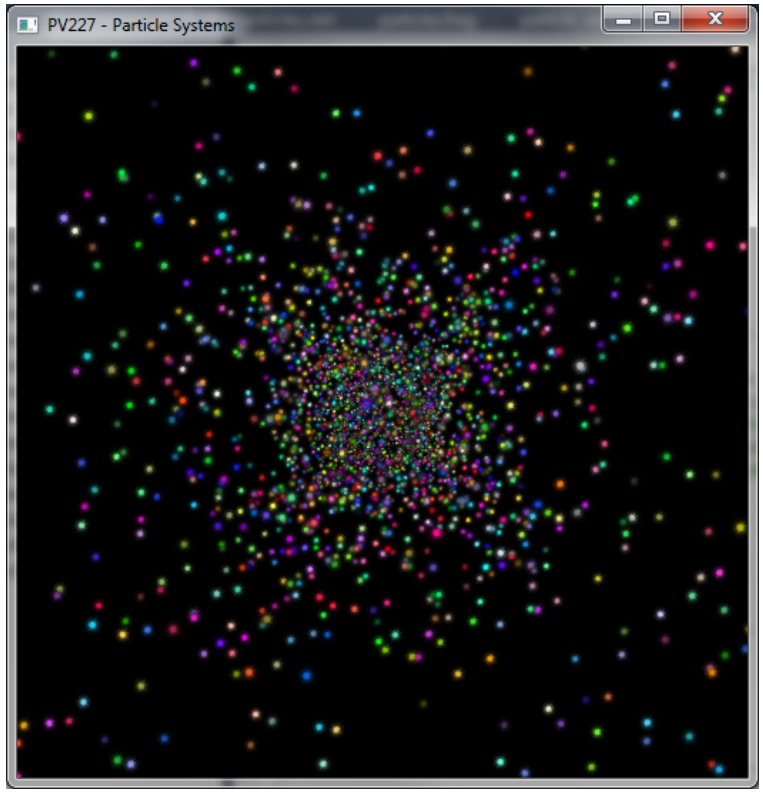


Figure: Particles representing stars in t = 0.

Iterative Particles

- render particles as points,
- update their position based on previous position.

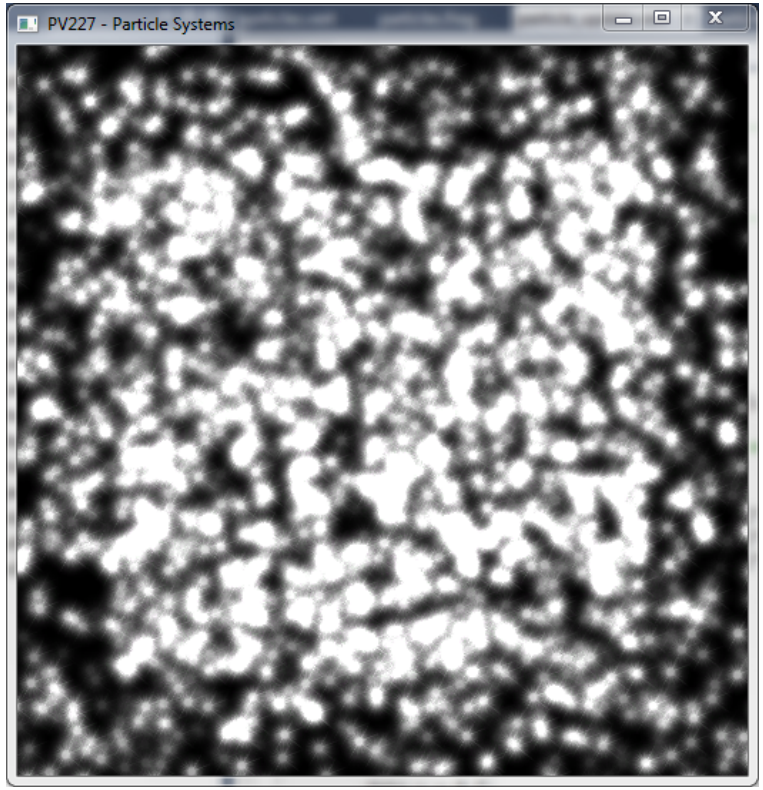


Figure: Emanating 64 $\times$ 64 particles in t = 1.f

Particles via Transform Feedback Buffer

- Firework with three types of particles
- updating position, velocity, TTL of particles

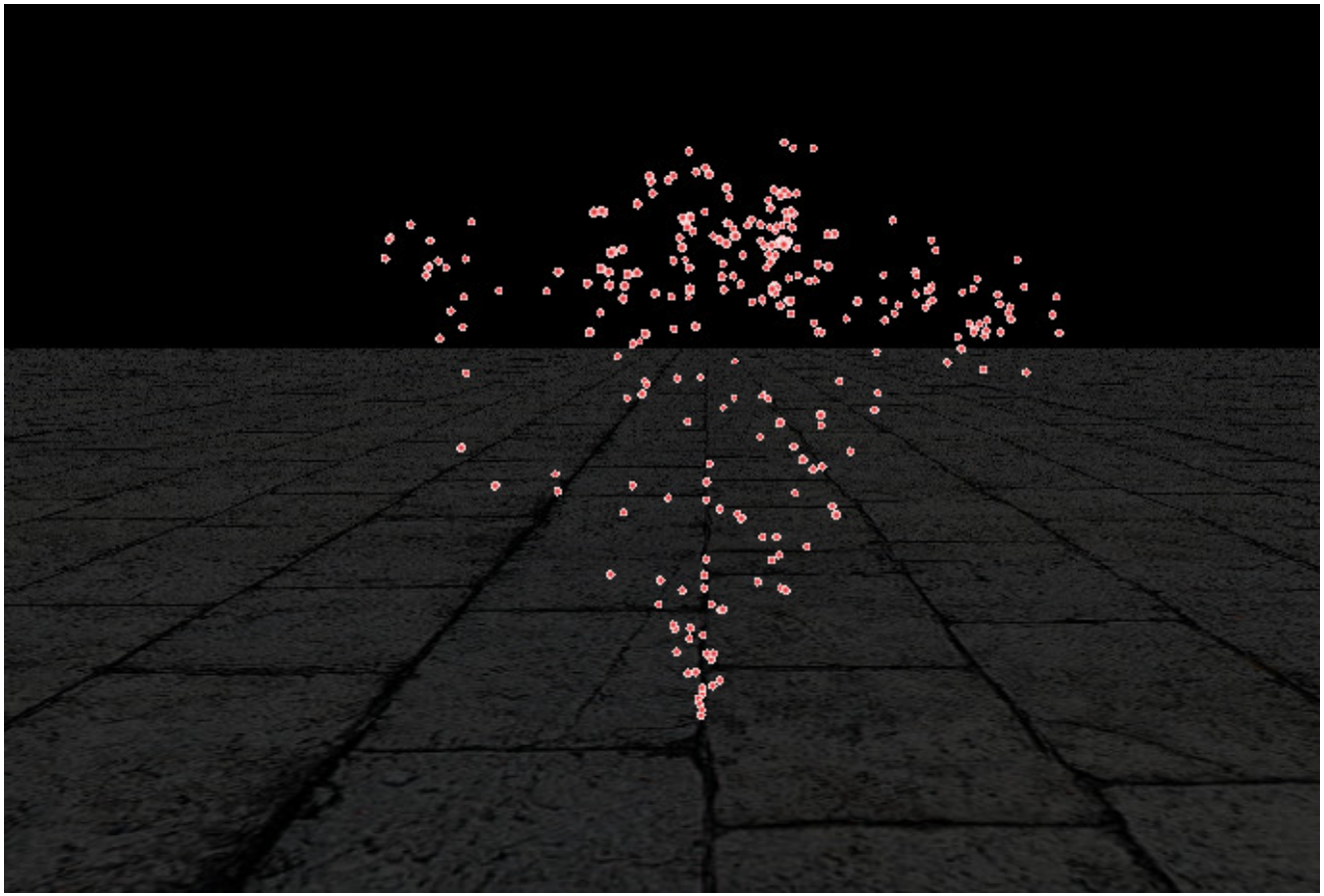


Figure: Shells and secondary shells (no visual distinction)