Game Developers Conference® | **March 9-13, 2010** | Moscone Center | San Francisco, CA

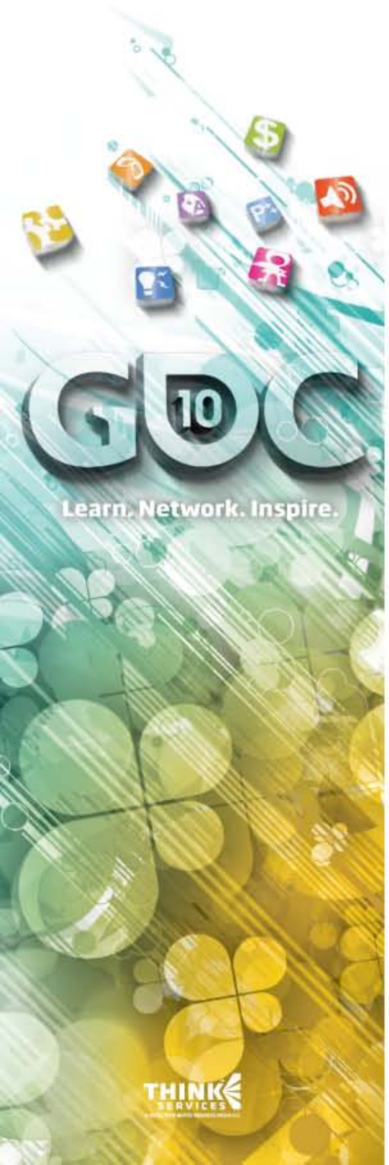
THINK SERVICES
A DIVISION OF UNITED BUSINESS MEDIA LLC

GDC 10

Learn. Network. Inspire.

www.GDConf.com

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com



Taking Fluid Simulation Out of the Box: Particle Effects in Dark Void

Joe Cruz – Airtight Games
Sarah Tariq - NVIDIA

Overview

- Motivation
- Demos
- Simulation
- Case Studies

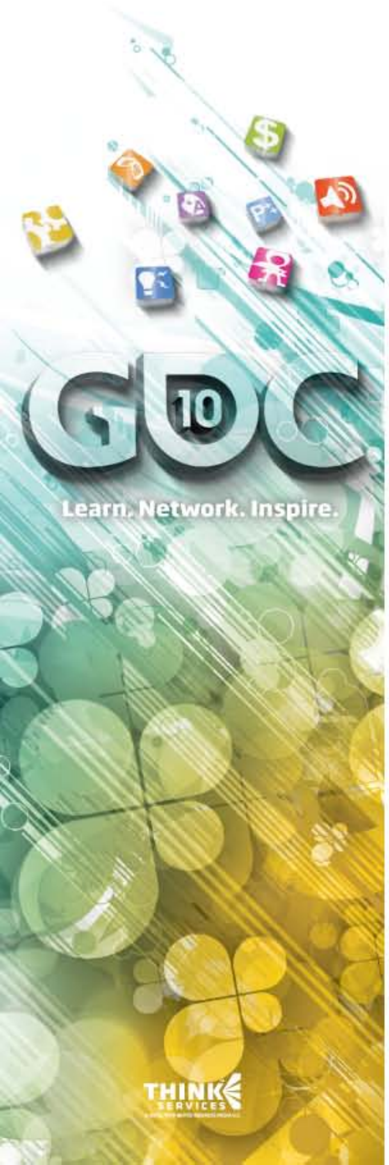
Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

Learn. Network. Inspire.

MOTIVATION

Motivation

- We had a number of important effects which would benefit from using fluid simulation
  - Smoke emitting from the jet pack of the character
  - Weapon effects
- Fluid simulation helped us improve the look of these effects and made the game more immersive and interactive

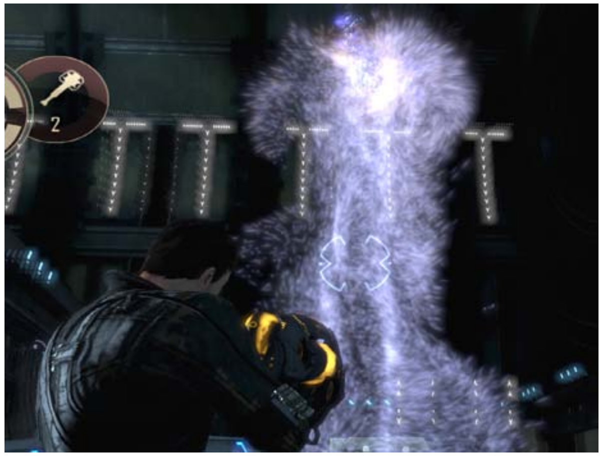


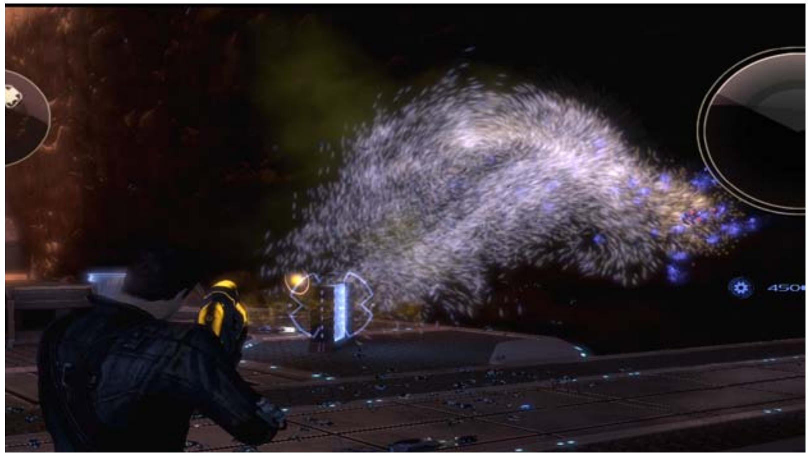
Fluid Simulation in Games

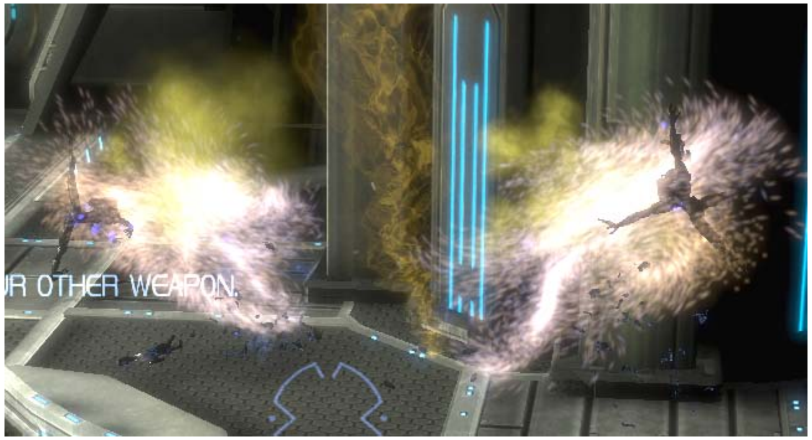
- Jos Stam presented stable fluids in 99 and fluid simulation for games at GDC 2003
- But it has taken a while for 3D fluid simulation to take hold in the game industry
    - Started seeing real fluid simulation in games only in the past few years
- Why?
    - Fluid simulation is expensive and does not look good at a very small resolution or in a very small region
    - Fluid simulation is viewed as hard to direct and control

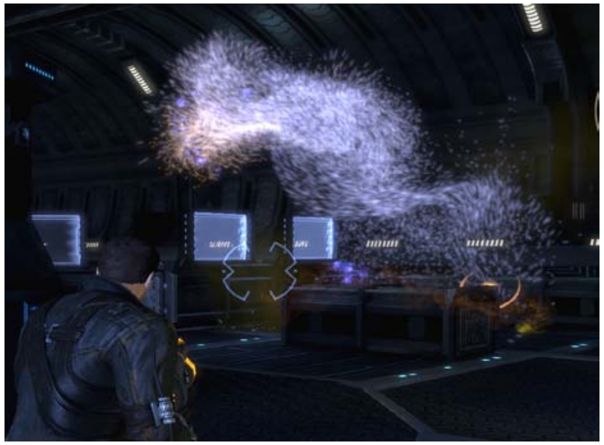
Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

Fluid Driven Effects In Dark Void





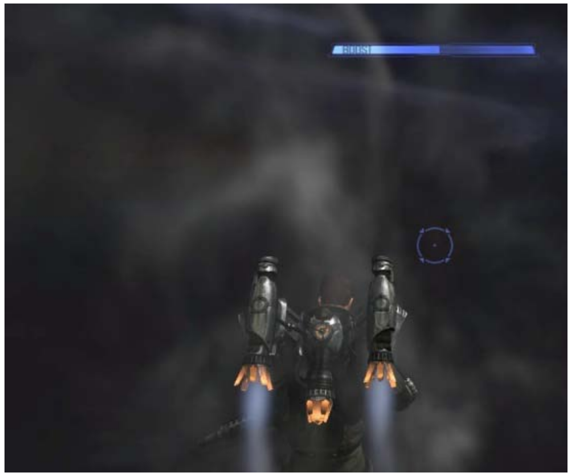


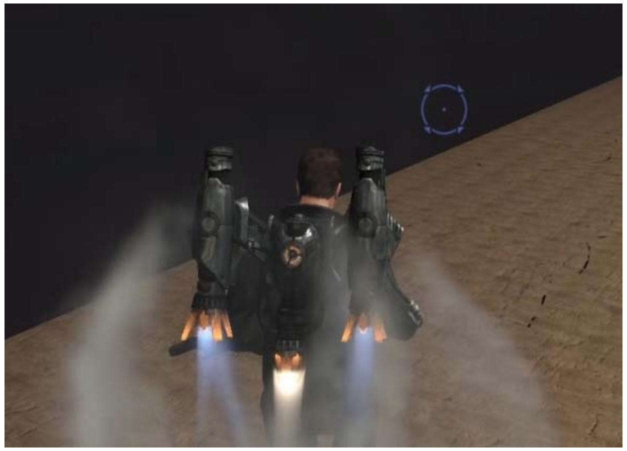
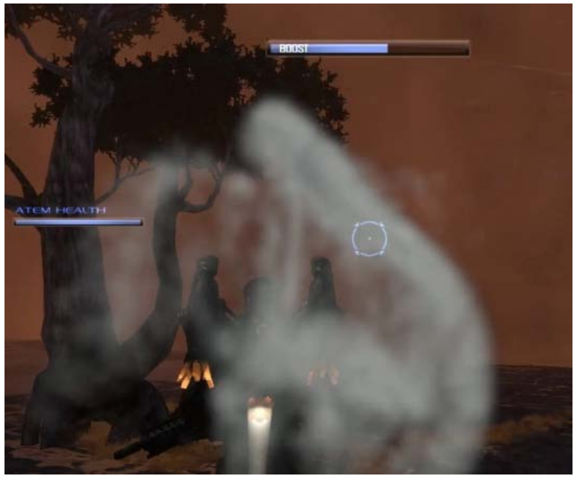


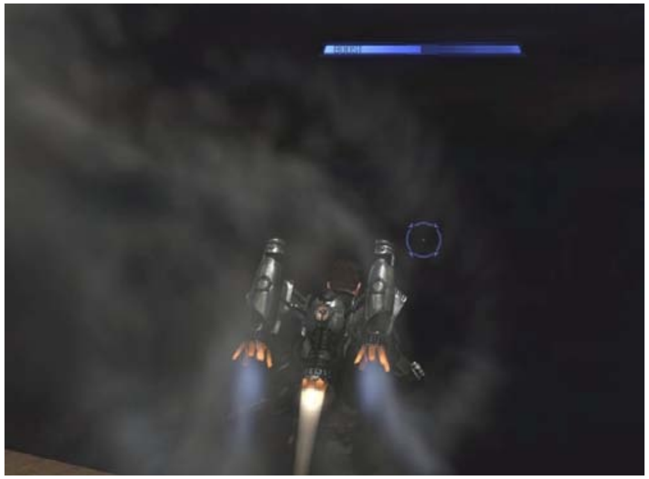
Learn. Network. Inspire.
THINK SERVICES

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

Fluid Driven Effects In Dark Void





GDC 10
Learn. Network. Inspire.
THINK SERVICES

DEMOS

BASICS OF FLUID SIMULATION

Describing a fluid

- A fluid with constant density and temperature is described by
  - **u** : its velocity field
  - **p** : its pressure field
- The task of a fluid solver is to compute **u**

Navier Stokes Equations

- Compact form

$$\frac{\partial u}{\partial t} = P\left(-\left(u \bullet \nabla\right)u + f\right)$$

Learn. Network. Inspire.
THINK SERVICES

Solving the equations

- Trying to solve

$$\frac{\partial u}{\partial t} = P\left(-(u \bullet \nabla)u + f\right)$$

- this is a PDE of the form

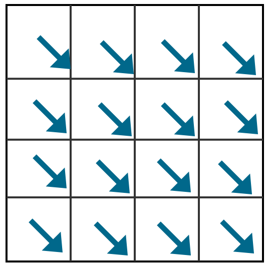
$$\frac{\partial x}{\partial t} = f(x,t)$$

- To use this equation we have to discretize the space, and then we can use for example Forward Euler*

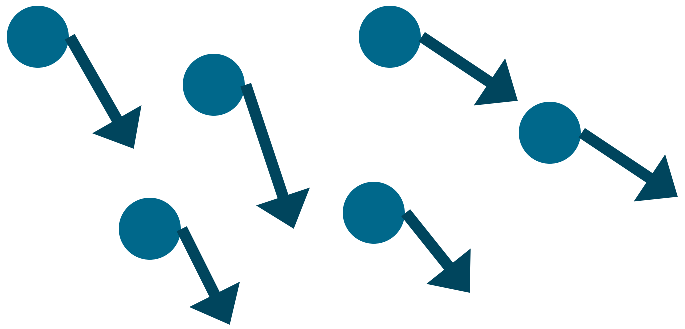
$$x^{n+1} = x^n + f\left(x^n, t^n\right)dt$$

Discretizing the space

- Grid Based (Eulerian)



- Particle based (Lagrangian)



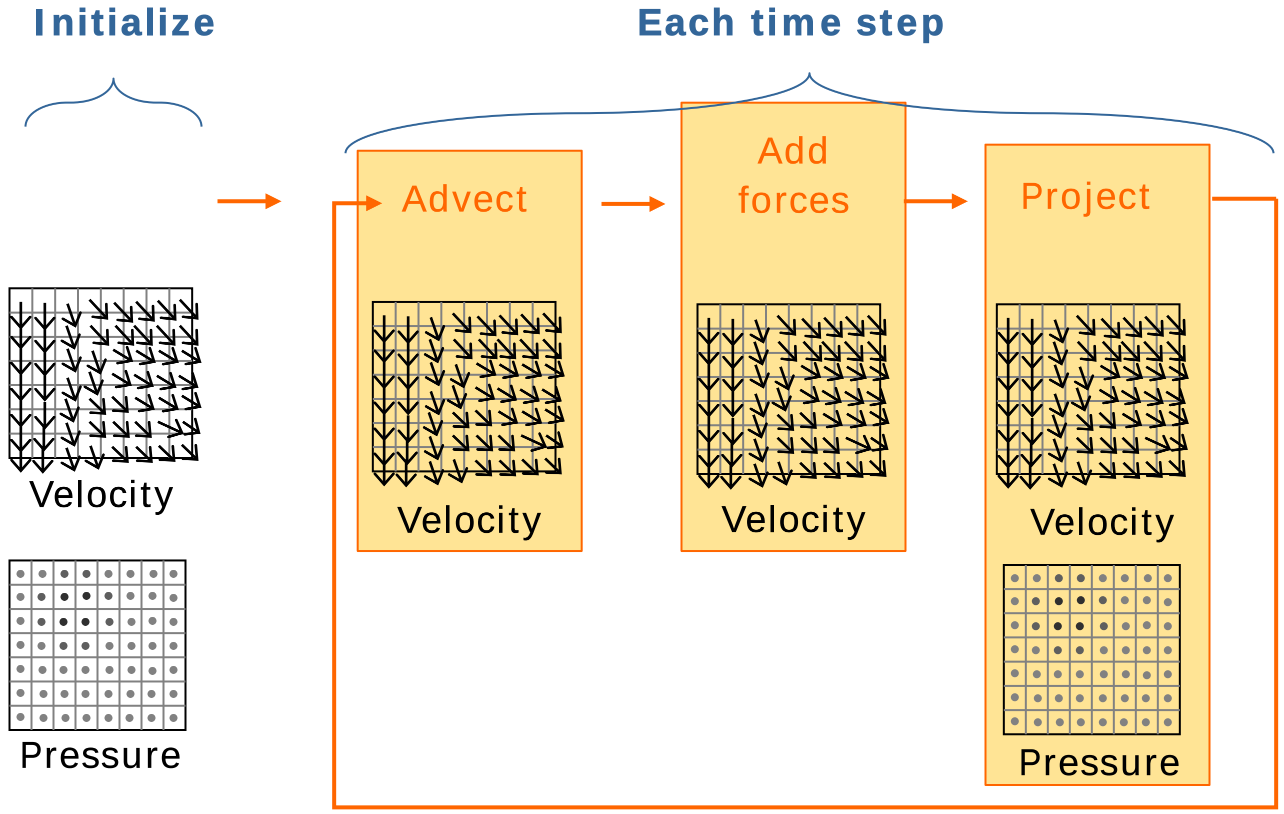
Learn. Network. Inspire.
THINK SERVICES

Eulerian vs. Lagrangian

- In Dark Void we explored using both SPH and Eulerian Simulation for different effects
  - Both approaches have their own advantages
- Particle based methods are not effective for simulating large regions of homogeneous fluids
  - Need an enormous amount of particles
- For the use cases that we had Eulerian Simulation was a better fit
  - Dense smoke
  - A lot of turbulent fine scaled motion was needed

Simple Eulerian Fluid Solver

$$\frac{\partial u}{\partial t} = \boxed{P}\left(\boxed{-(u \bullet \nabla)u} + \boxed{f}\right)$$



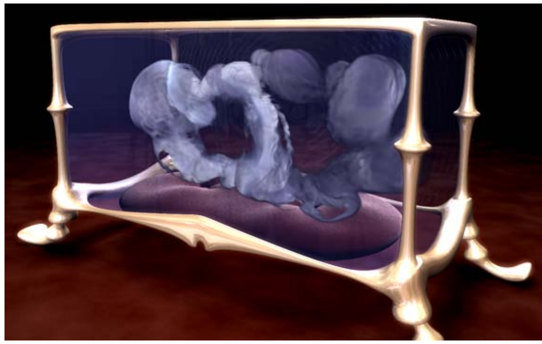
Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

Or how to live with the consequences of our choices

DEALING WITH EULERIAN SIMULATION

Eulerian Simulation Drawback: stuck in a box

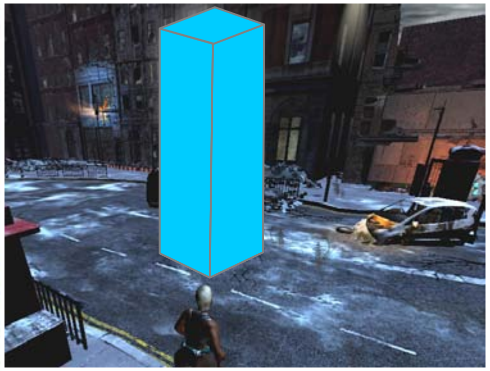
- Traditional issue associated with Eulerian fluid simulation is that simulation is confined to a finite rectilinear domain
- Thus it is hard to use in large environments with unconstrained motion

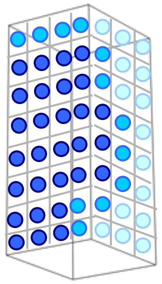

NVIDIA Fluid in a Box demo - 2006
Note that the fluid simulation is
stuck inside a stationary box

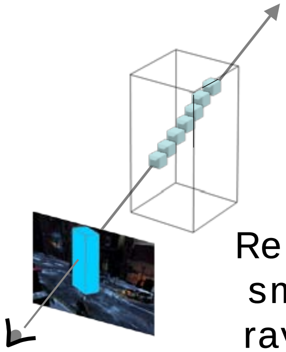
Typical Eulerian Simulation

Scene

Composite rendered smoke on top of scene


Decide where to place the smoke


Discretize the space and simulate


Render the smoke by raycasting the volume and accumulating simulated density

Moving Simulation out of the Box

- In order to allow our effects to move anywhere in the world we use a combination of

    - A moving fluid grid

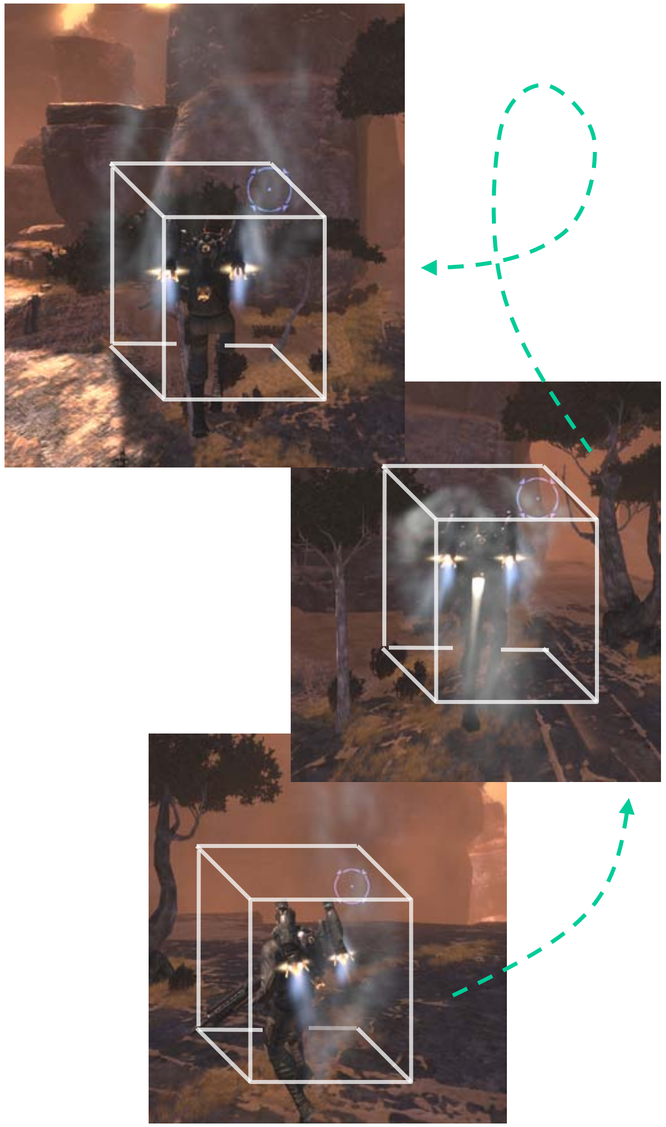
    - Fluid advected particles which are free to flow in and out of the Eulerian grid boundaries

NVIDIA APEX Turbulence demo - 2009

Learn. Network. Inspire.
THINK SERVICES

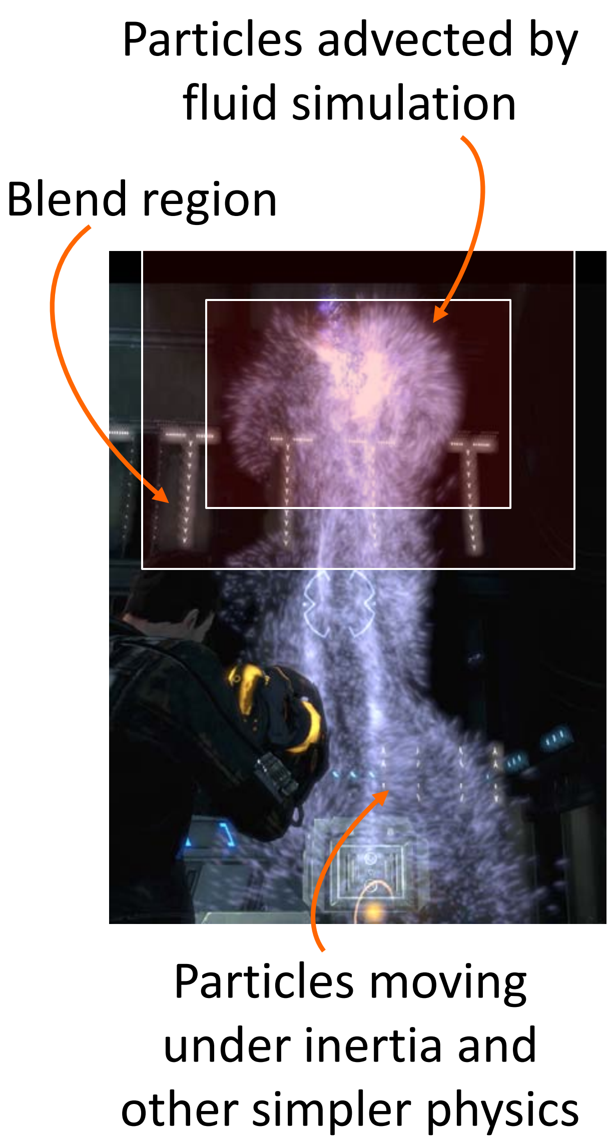
Moving the Simulation

- Our fluid grids can move to track an area of interest
- The simulation grid moves at the same speed as the object of interest
- Our assumption is that the interesting fluid dynamics are only happening in a bounded area



Using Particles

- When particles are inside the simulation grid they are pushed by the fluid
- When they are outside they follow other forces.


Particles advected by fluid simulation
Blend region
Particles moving under inertia and other simpler physics

Moving the reference frame

Simulating all the water in a pool might be too costly

=

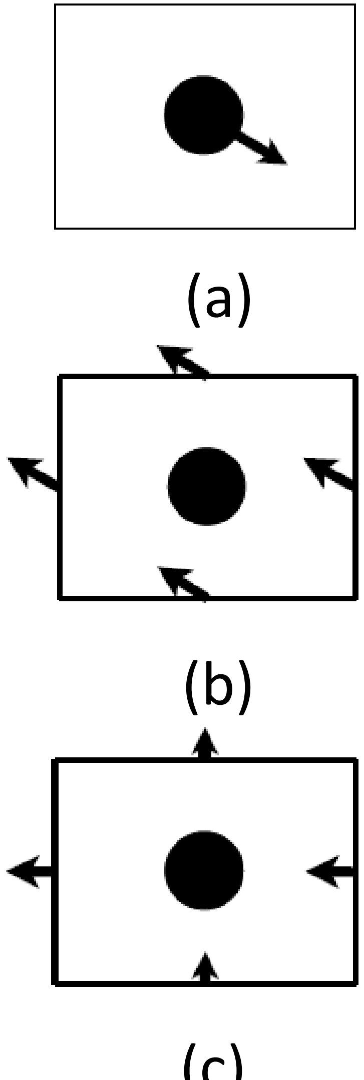
But the fluid dynamics are the same if instead we force an opposite flow past the body

- **Galilean Invariance**

* Images from Wikipedia under Creative Commons Attribution ShareAlike 3.0 license

Galilean Invariance

- Translating an object with velocity t
- Is the same as forcing a flow past the object with velocity –t
- In a coordinate frame translating with velocity t
- Same as specifying the flux across the boundaries



SIMULATION IN DARK VOID

Fluid Simulation Pseudo Code

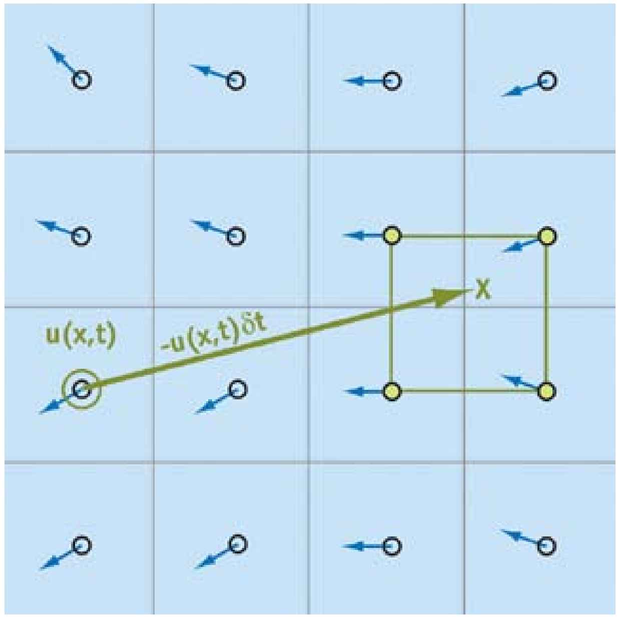
```
u = SelfAdvection(u)   (using MacCormack Scheme)

For i=1 to num_iop_iterations do

    Apply impulses to u
    Enforce solid boundary conditions on u

    Clear pressure
    Solve $\nabla^2 p = \nabla \bullet u$   (using multigrid method)
    $u = u - \nabla p$

end for
```

Step 1
Self Advection

- Advect the velocity forward based on the velocity field of the fluid

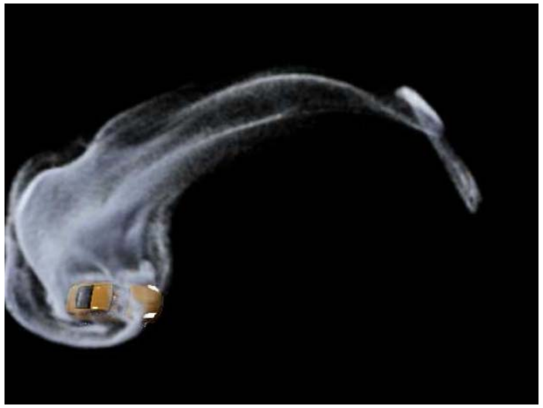
$$u^t(x) = u^{t-1}\left(\boxed{x - \Delta t * u^{t-1}(x)}\right)$$

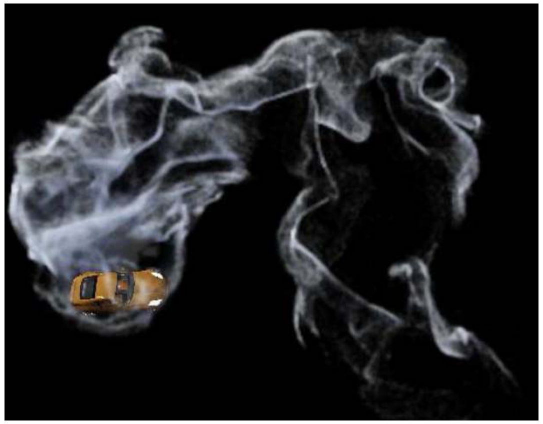
Semi Lagrangian advection



MacCormack Advection

- Basic semi-lagrangian advection is unconditionally stable, but it introduces unwanted numerical smoothing
- Use MacCormack scheme
  - Preserves detail
  - Stable for any time step
  - Trivially parallelizable and efficient on the GPU


Basic Semi-lagrange advection


MacCormack advection

Step 2

Enforce solid boundary conditions and apply impulses

- Treat internal boundary conditions using Iterated Orthogonal Projection framework
  - Lets us deal with internal solid objects
  - Also allows us to apply arbitrary impulses to the velocity prior to the projection step
    - We use these impulses for the jets
- Satisfying boundary conditions
  - Set the simulation cells that fall inside an obstacle to the relative velocity of that obstacle

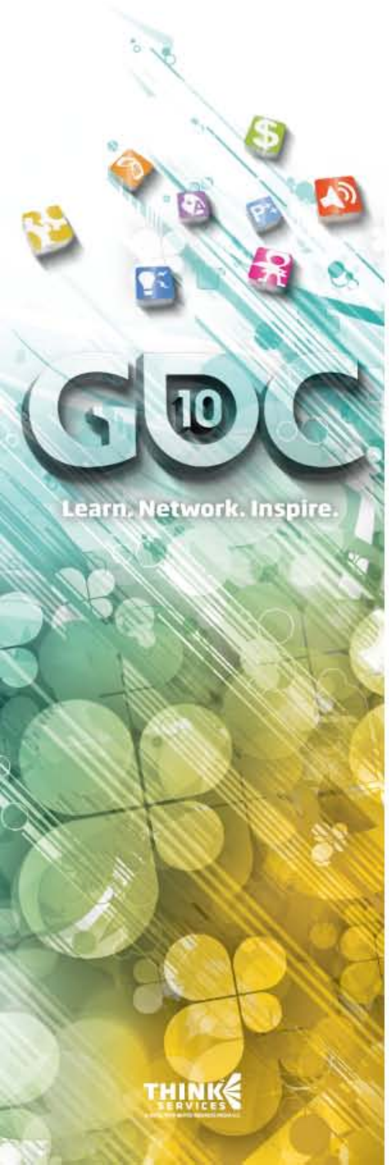
Step 3

Make the velocity field divergence free

- Enforced flux at boundaries should propagate instantaneously via pressure
  - Incompressible fluid = infinite speed of sound
- Therefore:
  - Pressure solver must fully converge
  - We cannot use Preconditioned Conjugate Gradient, since it does not converge fast enough
- We use Multigrid for this step

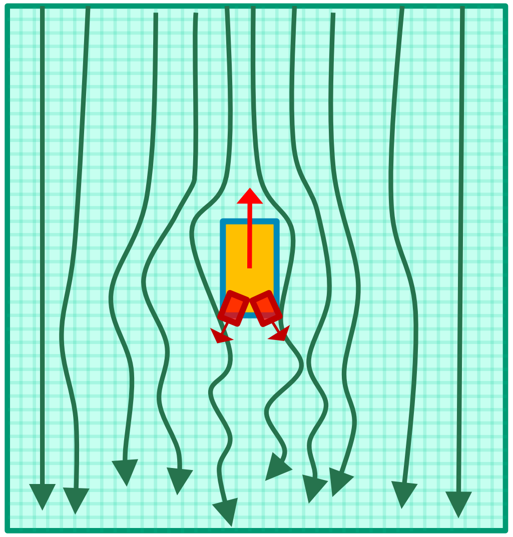
Particle Simulation

- Emitters in the scene add to one large particle buffer
    - Each emitter is allocated a separate region of this buffer
    - Emitters recycle dead particles
- For all the particles in this buffer we
    - Add forces based on the fluid grids
    - Add external forces for particles not falling in any grid
    - Update the particles, including moving them forward based on their velocity, increment their age, etc



Creating Turbulence

- As the object of interest moves it displaces air and leaves a turbulent wake behind it
- The object is represented as a collision obstacle in the simulation grid
- We also provide jets which can inject forces into the simulation
- By randomly varying the force direction and strength over time we can increase the turbulence in the system

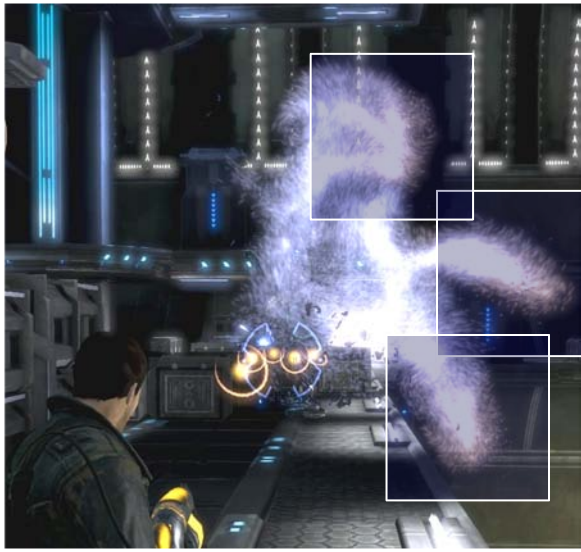


The fluid simulation grid.
The yellow object is the collision obstacle.
The red objects are the jets.
The green lines depict the flow.

Learn. Network. Inspire.
THINK SERVICES

Multiple Simulation Grids

- Fluid grids are simulated independently
- However, particles falling in overlapping regions of simulation grids aggregate forces from all of them


Three overlapping fluid simulation grids with their approximate boundaries. Particles in overlapping region get affected by all the relevant grids

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

PERFORMANCE

Cost

- Problem: fluid simulation is too costly
- Solution:

  Implement it on the **GPU**, which gives more than a 10x increase in performance

  - We used CUDA for all our simulation, flexible and easy to use

  Have a robust **LOD** and scalability system – more on this later

Performance numbers

Grid Size	Effect	Average Time
32x32x32	Disintegration gun	5 ms
48x48x48	Jetpack	13 ms

Computational time for entire simulation (fluid and particles).
Simulation is running on a GTX 285

Performance Bottlenecks

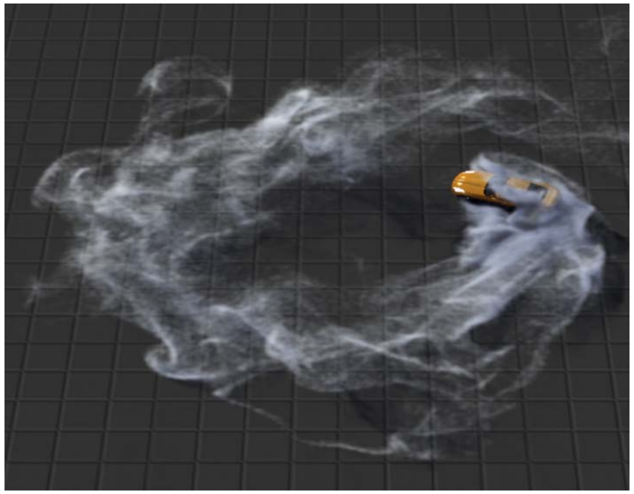
- Resolution of Simulation Grids
  - Lower resolution grids are faster in absolute terms
  - But lower resolution grids are also more inefficient – slower when measured in normalized cost per grid cell
- Number of separate grids
  - Cost of simulation increases sub-linearly with number of grids
  - particle advection cost also increases as grids overlap
- Number of active particles in the world
  - Each particle needs to sample velocity, move itself forward etc

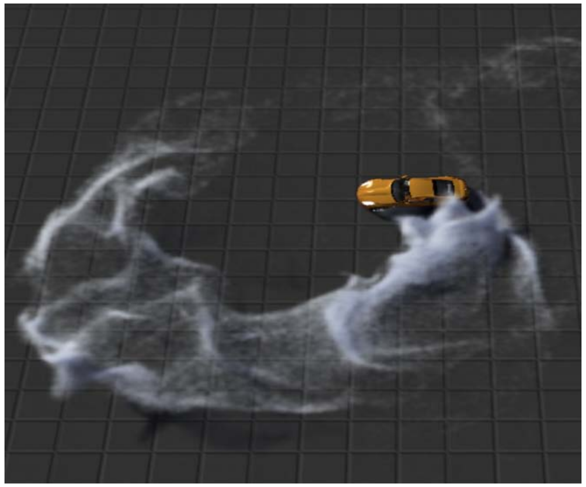
Learn. Network. Inspire.
THINK SERVICES

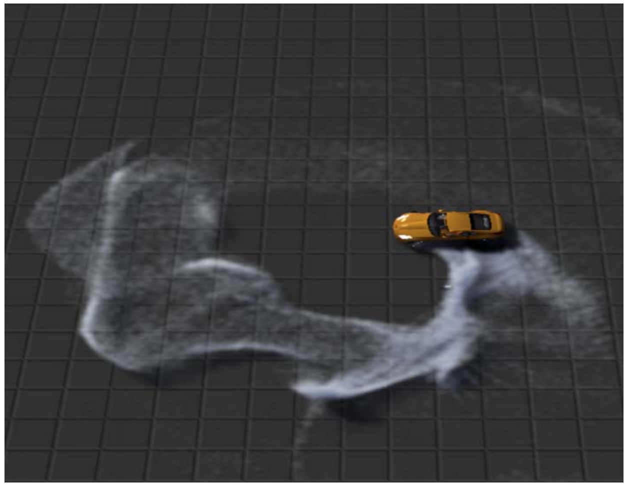
LOD options

- Change the size of simulation grids
  - Costly operation – need a hysteresis threshold to limit changing too often
- Simulation grids can skip an update every n frames
  - For example update the simulation every other frame to cut simulation time in half
  - Continue to sample velocity for particles every frame
- Control the max particle count for emitters
  - Emit fewer particles per frame based on computing resources available
  - Reduce the actual buffer size allocated to the emitter

Scaling the cost of the simulation


128x32x128 grid
500K particles
19 fps simulation
and rendering


64x16x64 grid
200K particles
25 fps simulation
and rendering


32x16x32 grid
100K particles
45 fps simulation
and rendering

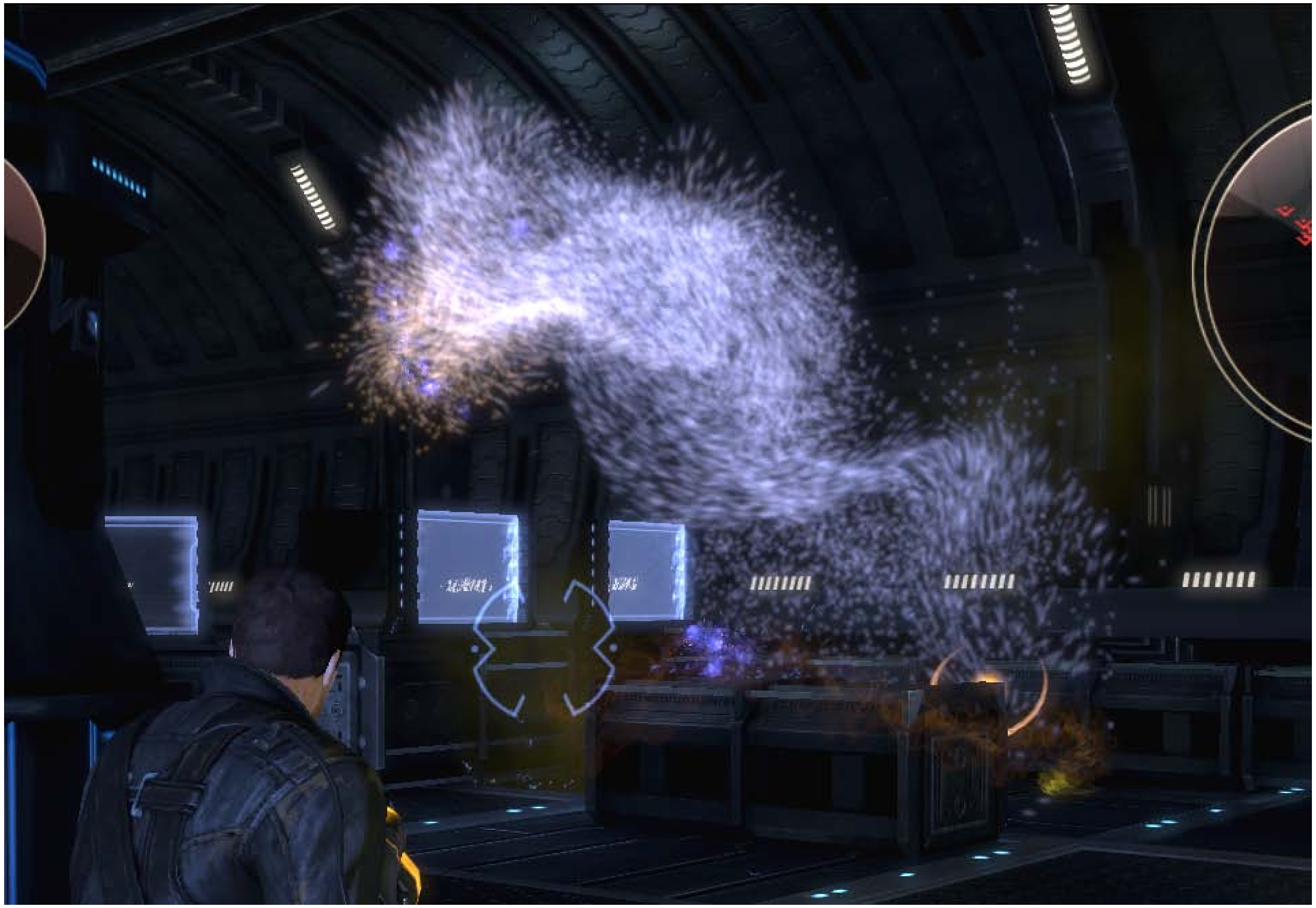
Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

Case St

MAKING THE EFFECTS

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

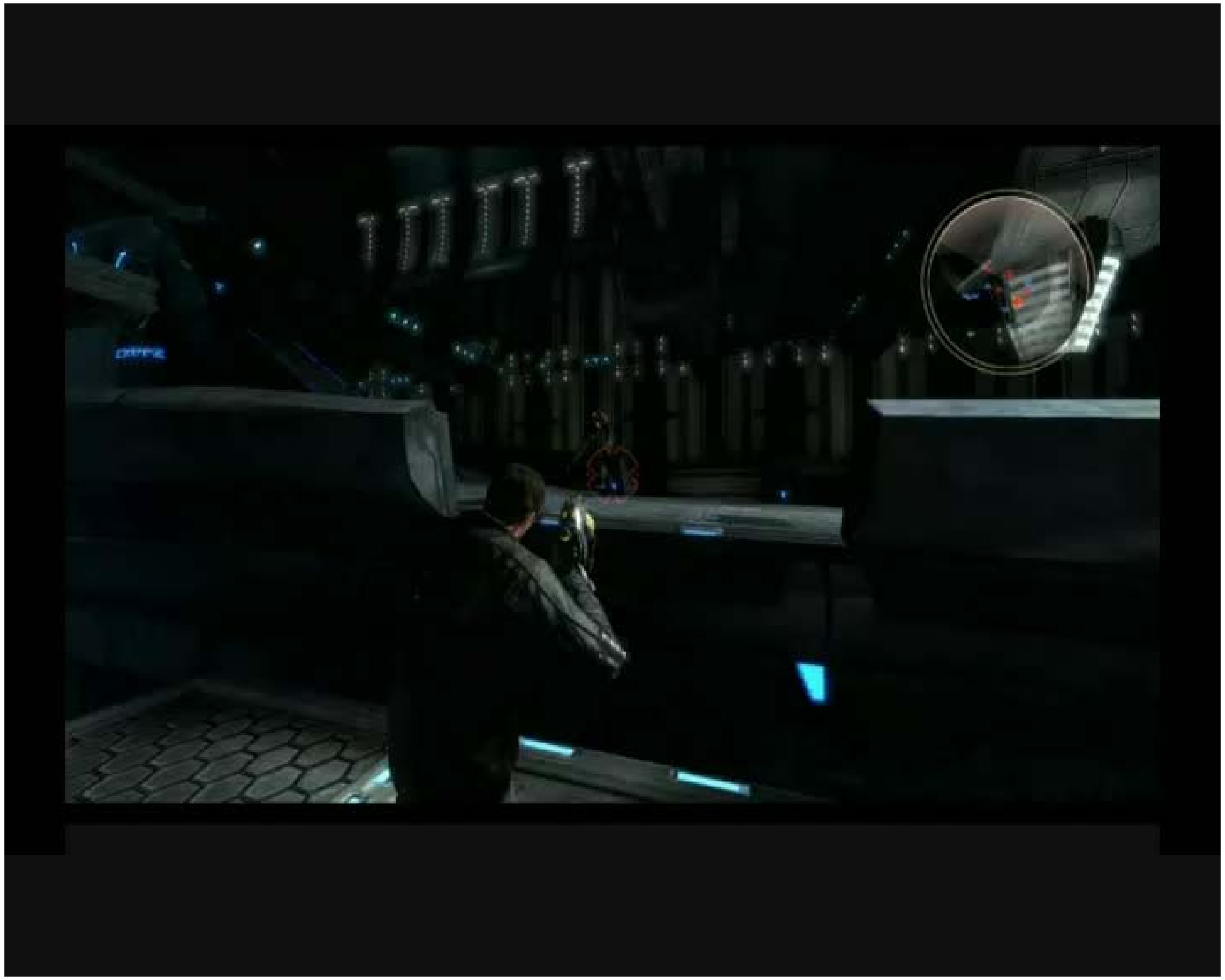
Disintegration Gun



GDC 10
Learn. Network. Inspire.
THINK SERVICES

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

Disintegration Gun

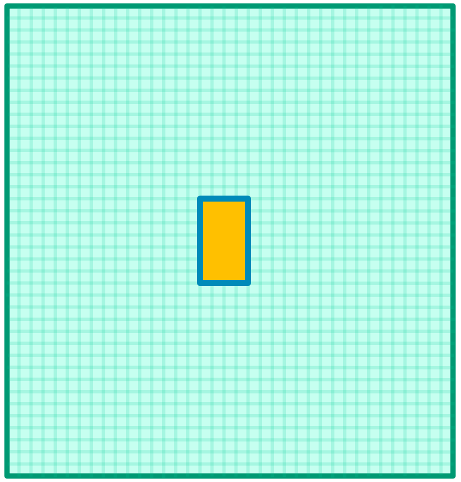


Disintegration Gun

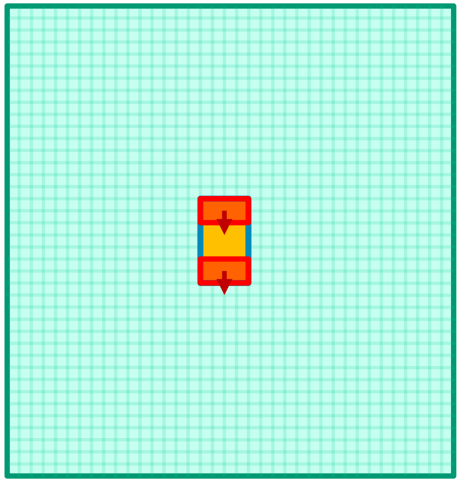
- Disintegration gun is one of the most powerful and interesting guns in the game
  - Its shots disintegrate anything they touch

- We wanted to experiment with a really visually interesting and memorable effect that was faithful to the characteristics of the gun

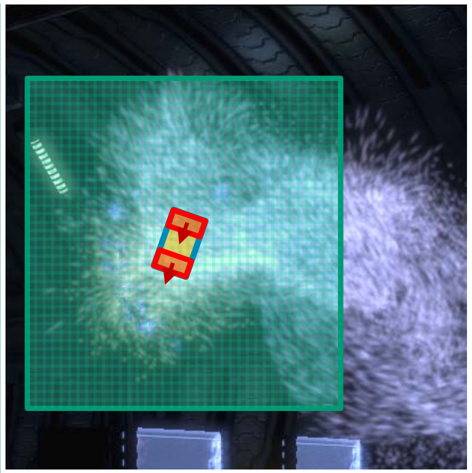
Disintegration Gun



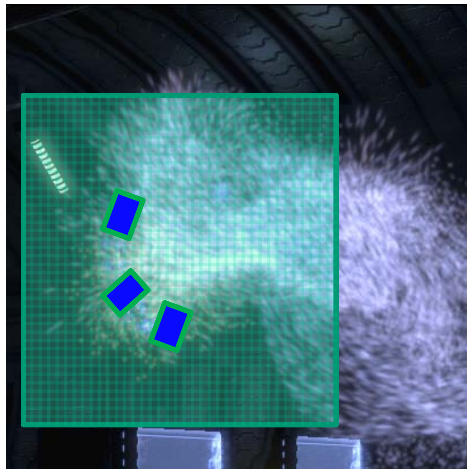
(a)
The fluid grid with the implicit collision obstacle in orange.



(b)
Two jet force emitters are added create extra turbulence. The direction of force is indicated by the arrows.



(c)
As the dying grunt moves the simulation grid (and the collision implicit and jets) move with it.
When the grunt rotates the collision implicit and jets rotate with it.



(d)
The emitters are placed and rotated independently. Here emitters are placed at the knees head and torso of the grunt.

Disintegration Gun Motion

- Simulation grid is centered on the dying grunt as it moves
  - 32x32x32 sized grid
  - Grid has a small collision obstacle at its center to create extra turbulence
  - As the grunt moves and rotates it imparts linear and angular forces into the simulation
- Emitters are tied to different body parts
  - The head, knee and torso
  - Each emitter emits 10,000 particles

Learn. Network. Inspire.
THINK SERVICES

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

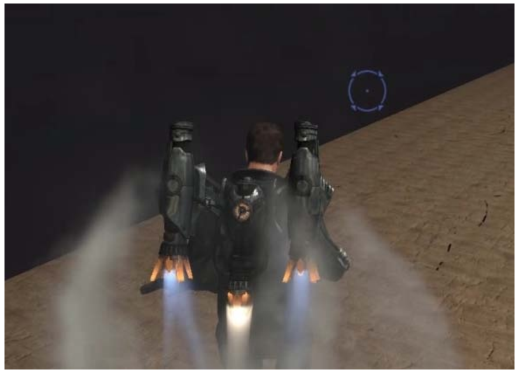
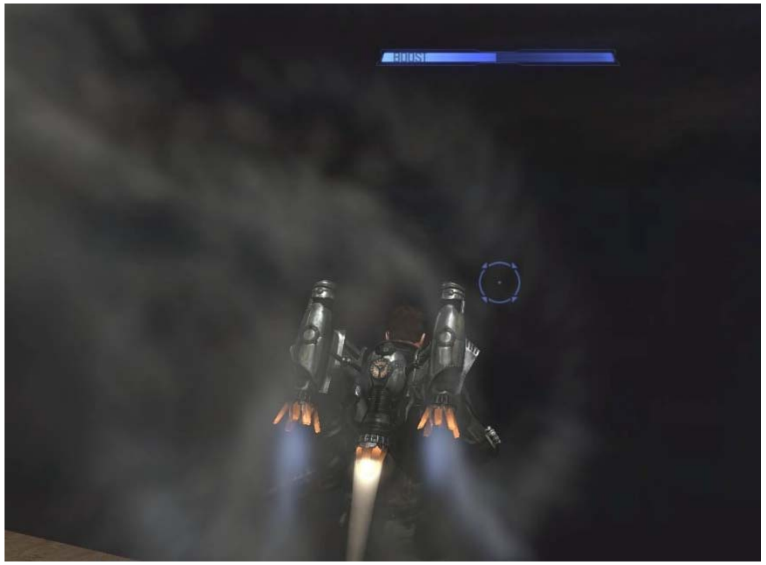
Disintegration Gun Motion

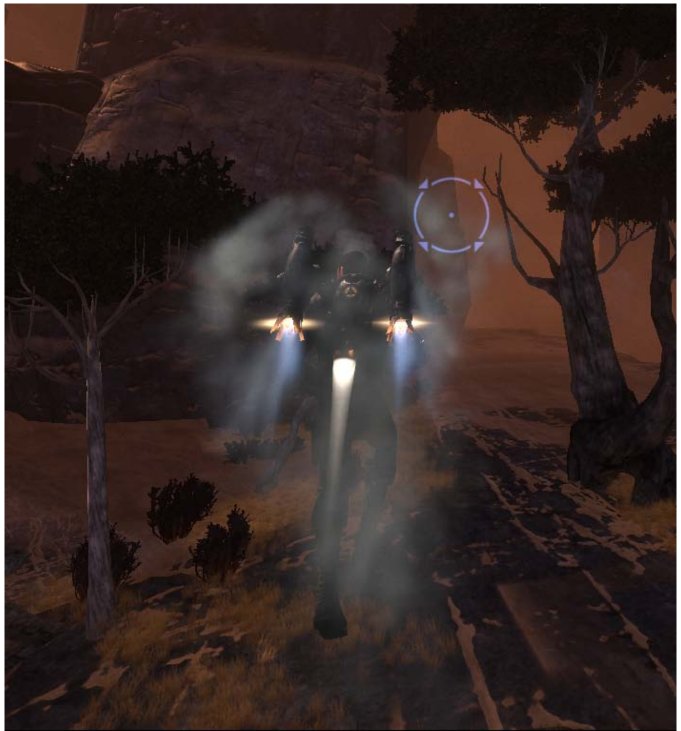
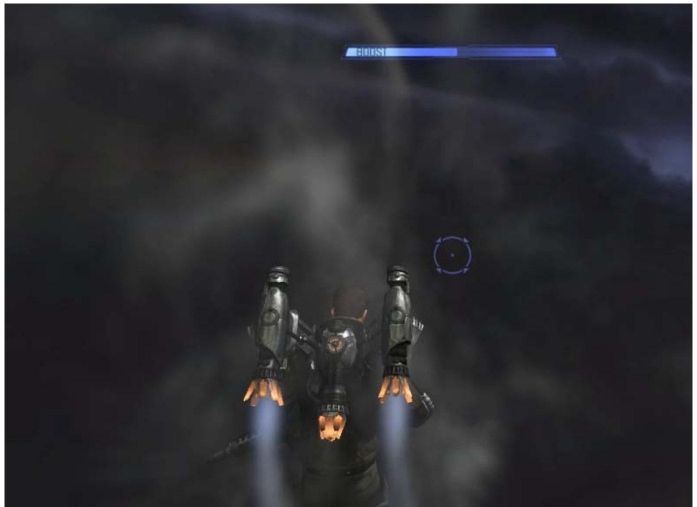
- Directional forces attached to same sockets as emitters
  - forces are smoothly randomized over time
  - Forces are also pulsed on and off in addition to the randomization
- Forces are turned off after the emitter, and emitter is turned off before the final effect finishes

Disintegration Gun Rendering

- Particles are rendered as camera facing quads
- Quads are stretched in the direction of motion to simulate motion blur
- Particles are rendered with additive blending
- We blend between different textures over the lifetime of a particle

Jet Pack

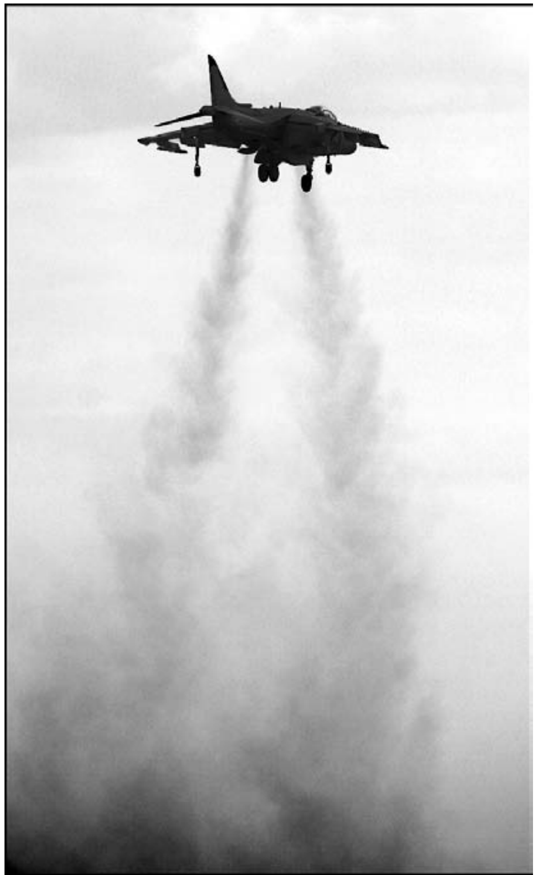



Real world references

GDC 10
Learn. Network. Inspire.
THINK SERVICES

Real World References

- Helicopter & jump jet hover vortices



GDC 10
Learn. Network. Inspire.
THINK SERVICES

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

The Jet Pack – in game

GDC 10
Learn. Network. Inspire.
THINK SERVICES

Jet Pack

- We needed to mimic a helicopter turbine wash as a jetpack hover
- Simulation Grid attached to player's jetpack
  - The character is represented as a collision obstacle centered in the grid
  - Grid moves at same velocity as character
  - Grid is 48x48x48 cubed

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

Jet Pack

- Emitters placed at jetpack nozzles
  - Attached to an artist defined sockets
  - Emitters have between 15000 to 24000 particles each
- Force jets around emitters
  - Also attached to socket

Jet Pack

- Actual implementation in-game
- Challenges were:
  - Player flight motion relative to world was tuned for gameplay, not realism
  - Character animation relative to camera was added to give a heightened perception of speed
- Result: initial turbulence simulation would show unrealistic results

Jet Pack: Solutions

-The challenge was to keep as many particles as possible within the turbulence grid, thereby maximizing the visual impact of the turbulence simulation, despite very rapid changes in velocity of the player

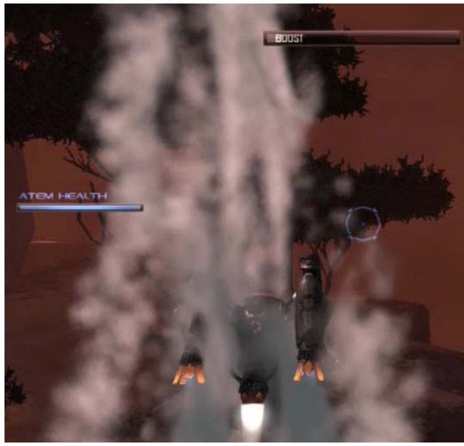
-balancing grid size, direction and magnitude of jet forces per effect were key parameters in helping the artist solve this production problem

The Look

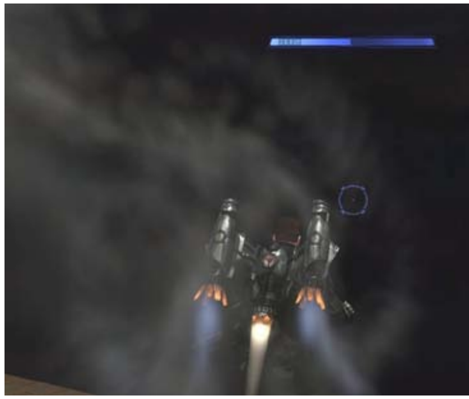
- Difficulty creating the final look of the jetpack

  Initial iteration showcased the simulation and rendering but was not consistent with what a jetpack should look like

  It looked more like a steam engine ☺



Initial look

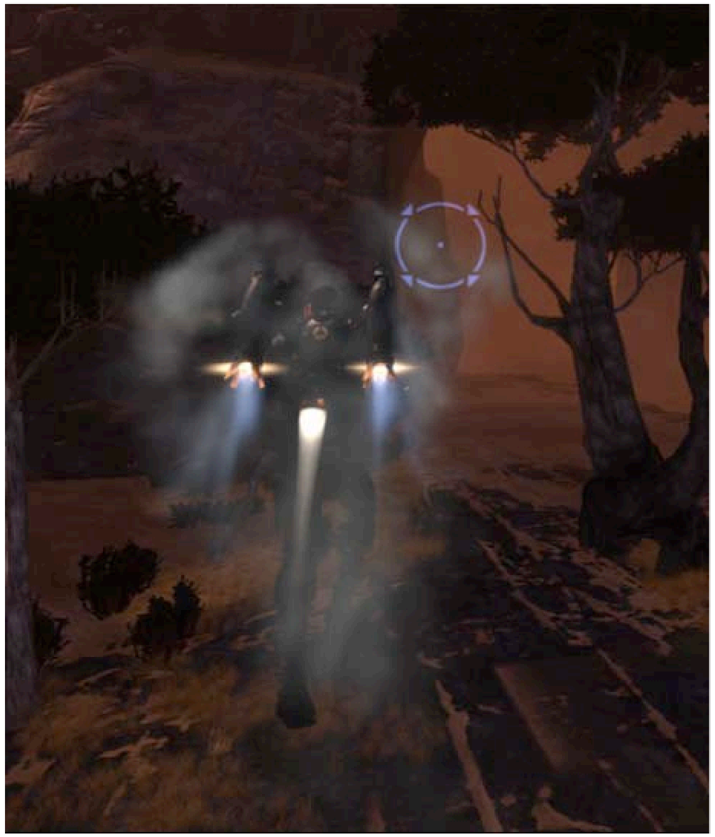

Final look

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

The Look

- We eventually settled on having a much more transparent blue heat effect
- Thick smoke is emitted whenever there is a change in applied power (i.e. the jetpack turns on or off)
- When the character lands the jetpack shuts down and emits a final burst of smoke

Real vs. Simulated



Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

CONCLUSION

Learn. Network. Inspire.

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

Takeaway

- Detailed interactive 3D fluid simulation is
  - Feasible in a game's budget today
  - Versatile and art direct-able
  - Scalable

GDC 10 Learn. Network. Inspire.
THINK SERVICES

Additional Information

- APEX Turbulence:
- Used for all the fluid simulation in Dark Void
- http://developer.nvidia.com/object/apex_turbulence.html

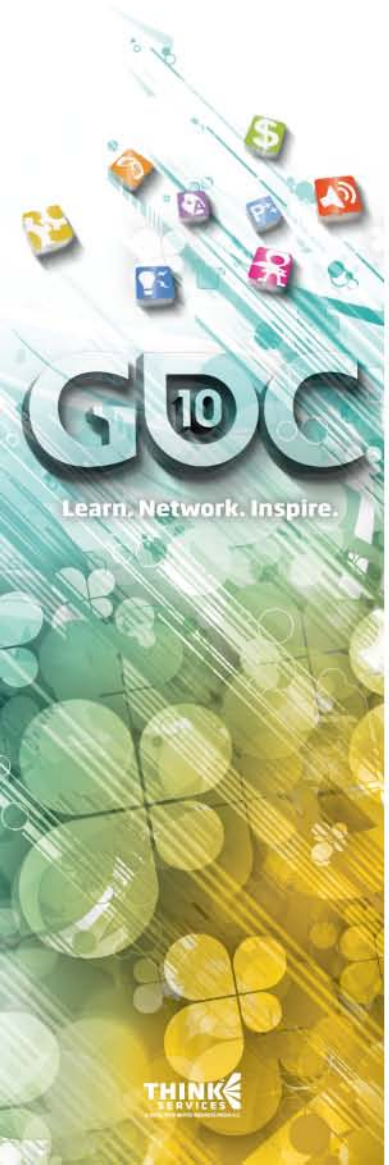
- Interactive Fluid-Particle Simulation using Translating Eulerian Grids, I3D 2010. http://www.jcohen.name/papers/Cohen_Interactive_2010.pdf

- Fluid simulation on GPU basics, GDC 2007
- http://developer.download.nvidia.com/presentations/2007/gdc/RealTimeFluids.pdf

GDC 10
Learn. Network. Inspire.
THINK SERVICES

Acknowledgements

- Lots of thanks to
  - Bryan Duduash, Neil Nafus, Dane Johnston, Johnny Costello
- Airtight Games
- Capcom
- NVIDIA



Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

Learn. Network. Inspire.

APPENDIX

Game Developers Conference®
March 9-13, 2010
Moscone Center
San Francisco, CA
www.GDConf.com

MacCormack advection

- Basic semi lagrange advection is a single forward advection step (denoted below by *A*)
- MacCormack Advection*:

$$\textit{forward} \quad \textit{step}: \quad \hat{\phi}^{n+1} = A\left(\phi^{n}\right)$$

$$\textit{backward} \quad \textit{step}: \quad \hat{\phi}^{n} = A\left(\overset{\frown}{\phi}^{n+1}\right)$$

$$\textit{estimate} \quad \textit{error}: \quad e = \left(\hat{\phi}^{n} - \phi^{n}\right) \big/ 2$$

$$\textit{final} \quad \textit{estimate}: \quad \phi^{n+1} = \hat{\phi}^{n+1} + e$$

$$\textit{clamp} \quad \phi^{n+1}:$$

$$\begin{aligned} \phi_{\max} &= \max\left(\phi^{n}, \phi^{n+1}\right) \\ \phi_{\min} &= \min\left(\phi^{n}, \phi^{n+1}\right) \end{aligned}$$

$$\textit{if}\,(\phi^{n+1} > \phi_{\max})\,\textit{or}\,(\phi^{n+1} < \phi_{\min}) \quad \phi^{n+1} = \hat{\phi}^{n+1} + 0.5 * e$$

$$\textit{clamp} \quad \phi^{n+1} \quad \textit{between} \quad \min(\phi^{n+1}, \phi_{\max}) \quad \textit{and} \quad \max(\phi^{n+1}, \phi_{\min})$$