Intuitive Rocket Science: The Game: The Presentation

Mark Henderson

&

Benjamin Kantor

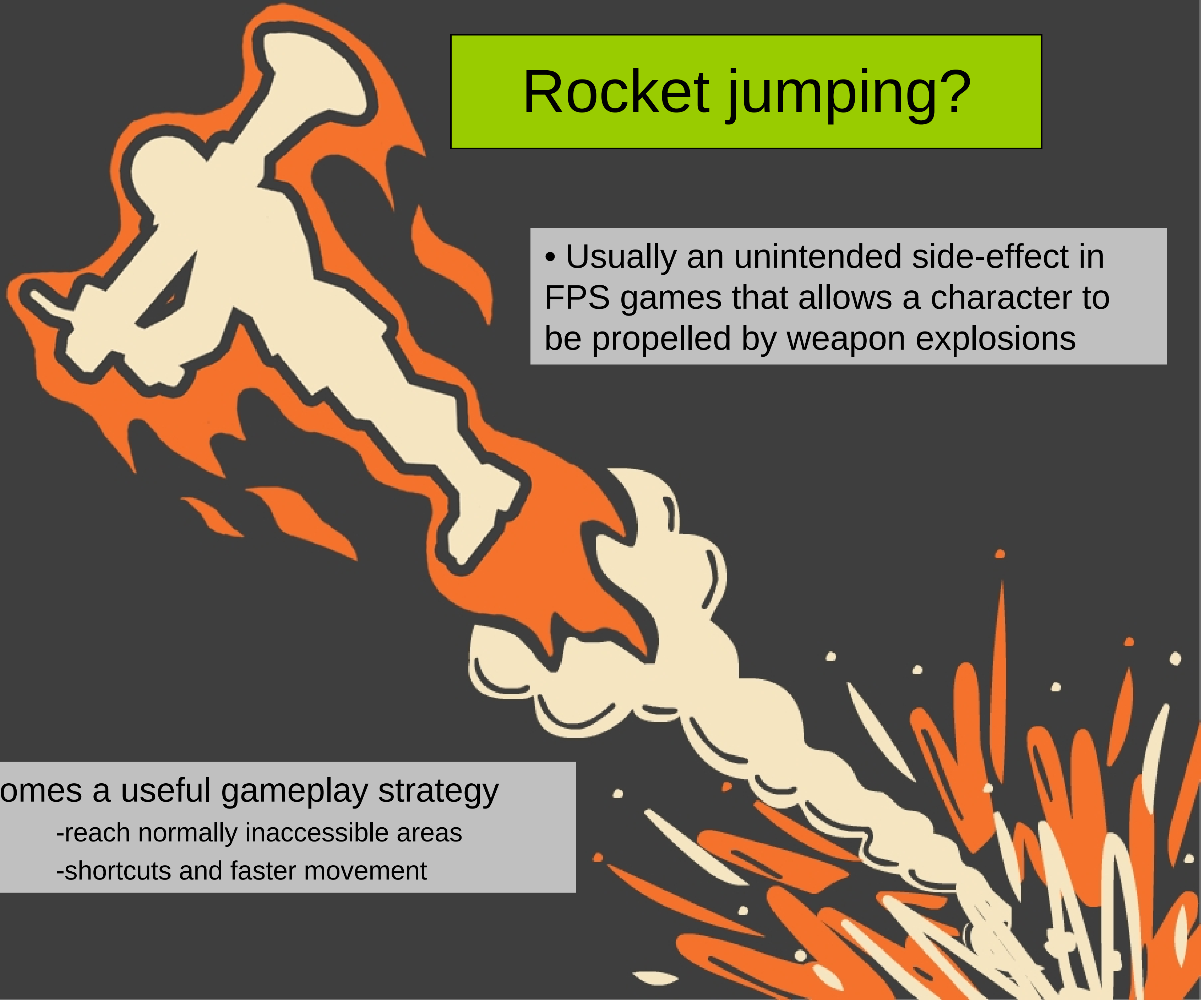
Gamebryo Element™ Academic Edition

Overview

- Background
- Methods
- Results
- Tools & Languages
- Contributions
- Future Directions

What is it?

- A game combining rocket jumping and astronomical scale gravity action!
- Built with C++, Gamebyro, and PhysX



Rocket jumping?

- Usually an unintended side-effect in FPS games that allows a character to be propelled by weapon explosions

- Becomes a useful gameplay strategy
  - -reach normally inaccessible areas
  - -shortcuts and faster movement

Team Fortress 2 Example

- ppt_images\Team Fortress 2 Demoman sticky nade jump.mov

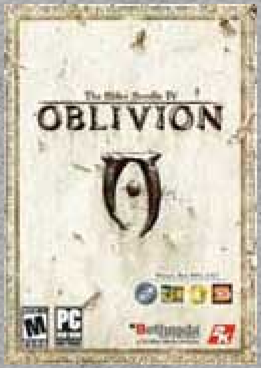
•Also special modded “jump maps”

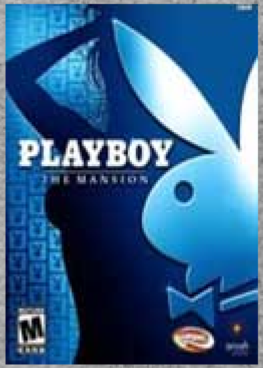
Super Mario Galaxy

ppt_images\Super Mario Galaxy trailer.mov

Emergent Gamebryo

- Popular game engine for PC, XBox360, PS3, and Wii
- Used to develop a wide variety of games

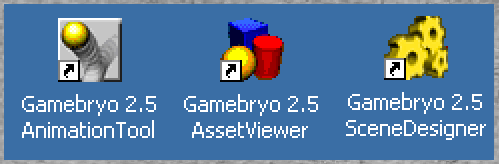




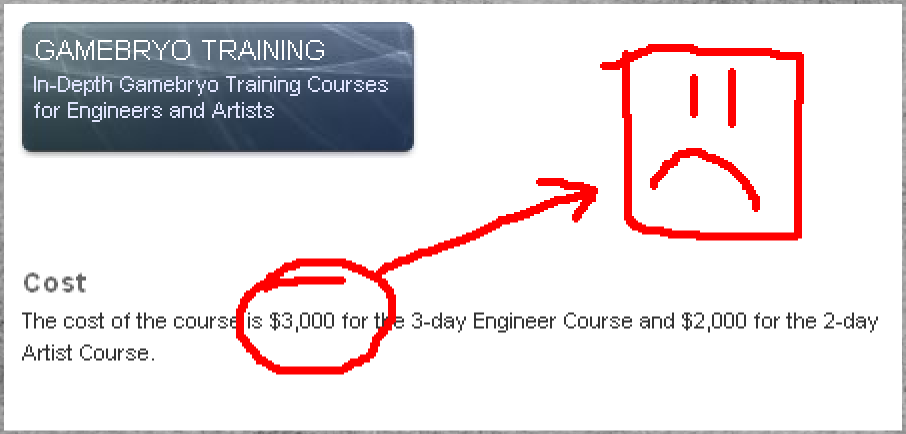
- Unfortunately not open source
  – Gamebryo 2.5 Evaluation: watermark and license restrictions

Emergent Gamebryo

- Object-oriented C++ 3D game engine
- Also includes Maya plug-in and 3 utilities:



- 3 filetypes: .NIF, .KF, .KFM
  - Geometry, Keyframes, Keyframe Manager



GAMEBRYO TRAINING

In-Depth Gamebryo Training Courses for Engineers and Artists

Cost

The cost of the course is $3,000 for the 3-day Engineer Course and $2,000 for the 2-day Artist Course.

Asset & Animation Pipeline

Maya

|

Sequence Editor

|

Export .NIF/.KFM

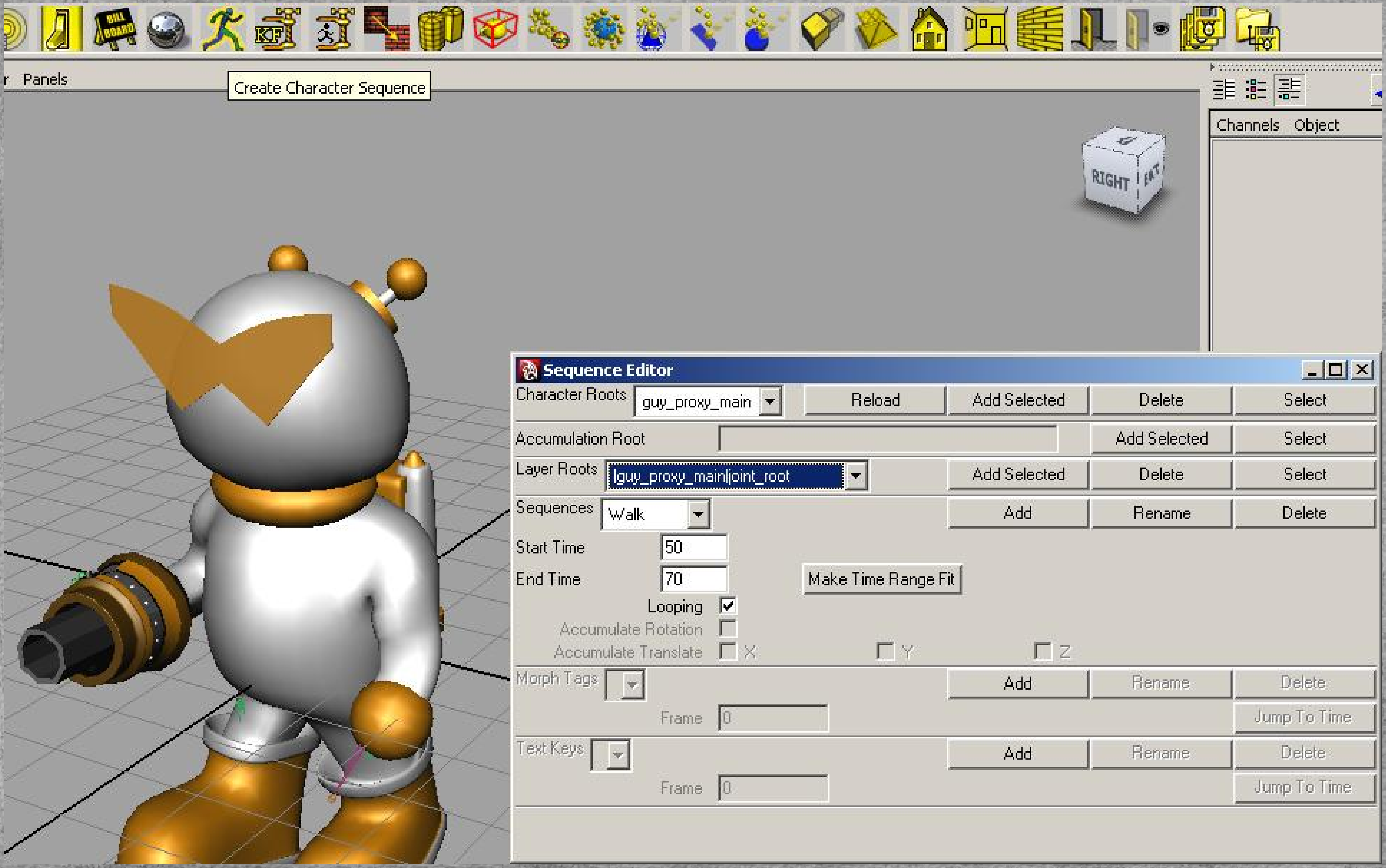
|

Animation Tool

|

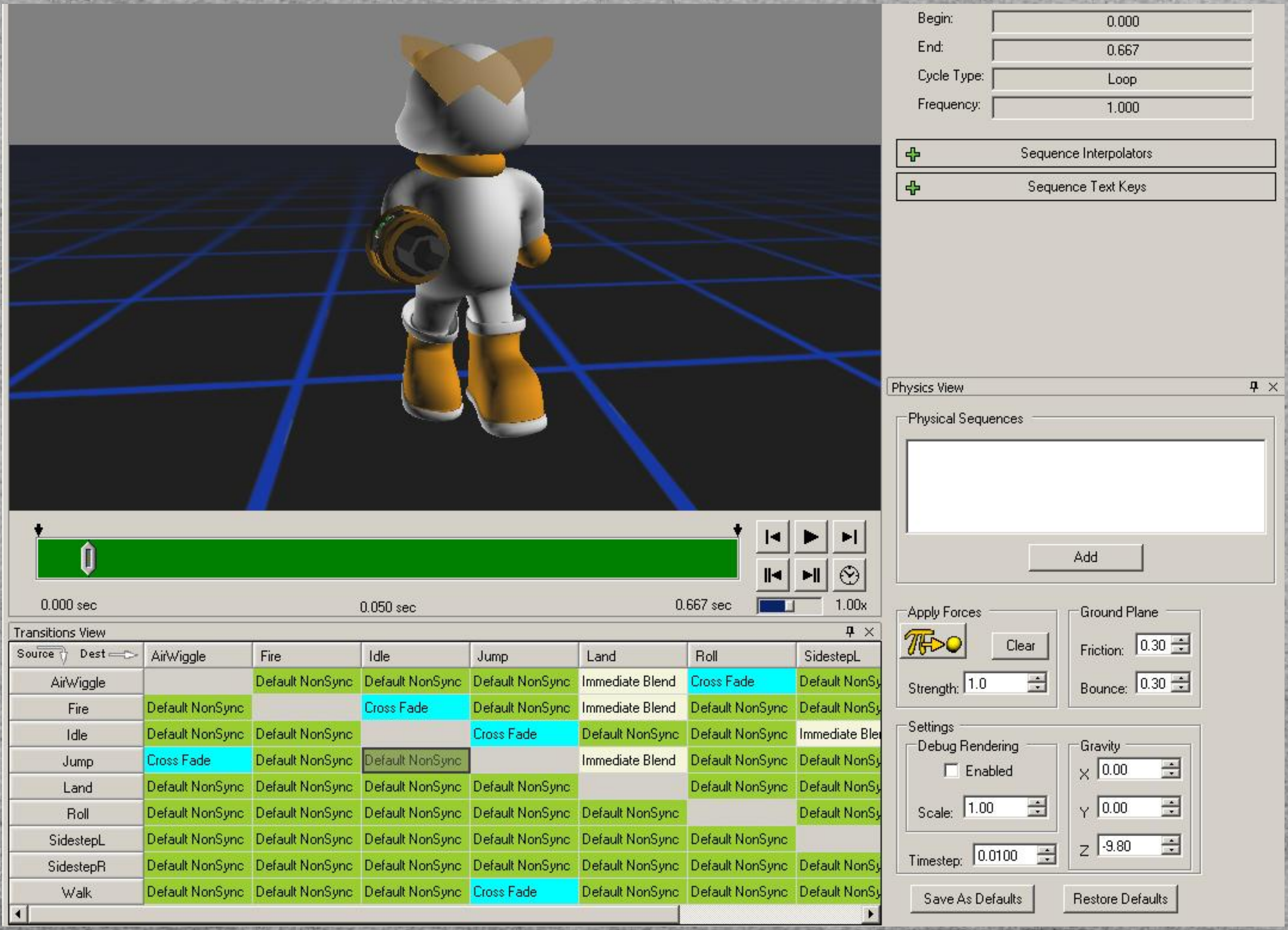
NiActorManager

Maya & Gamebryo Plug-in



Character Roots	guy_proxy_main	Reload	Add Selected	Delete	Select
Accumulation Root				Add Selected	Select
Layer Roots	|guy_proxy_main|joint_root		Add Selected	Delete	Select
Sequences	Walk		Add	Rename	Delete
Start Time	50				
End Time	70	Make Time Range Fit			
Looping	✓				
Accumulate Rotation	☐				
Accumulate Translate	☐ X	☐ Y	☐ Z		
Morph Tags			Add	Rename	Delete
Frame	0				Jump To Time
Text Keys			Add	Rename	Delete
Frame	0				Jump To Time

Animation Tool



Begin: 0.000
End: 0.667
Cycle Type: Loop
Frequency: 1.000

Sequence Interpolators
Sequence Text Keys

0.000 sec   0.050 sec   0.667 sec   1.00x

Physics View

Physical Sequences

Add

Apply Forces
Clear
Strength: 1.0

Ground Plane
Friction: 0.30
Bounce: 0.30

Settings
Debug Rendering
☐ Enabled
Scale: 1.00
Timestep: 0.0100

Gravity
X 0.00
Y 0.00
Z -9.80

Save As Defaults
Restore Defaults

Transitions View

Source / Dest	AirWiggle	Fire	Idle	Jump	Land	Roll	SidestepL
AirWiggle		Default NonSync	Default NonSync	Default NonSync	Immediate Blend	Cross Fade	Default NonSy
Fire	Default NonSync		Cross Fade	Default NonSync	Immediate Blend	Default NonSync	Default NonSy
Idle	Default NonSync	Default NonSync		Cross Fade	Default NonSync	Default NonSync	Immediate Blen
Jump	Cross Fade	Default NonSync	Default NonSync		Immediate Blend	Default NonSync	Default NonSy
Land	Default NonSync	Default NonSync	Default NonSync	Default NonSync		Default NonSync	Default NonSy
Roll	Default NonSync	Default NonSync	Default NonSync	Default NonSync	Default NonSync		Default NonSy
SidestepL	Default NonSync	Default NonSync	Default NonSync	Default NonSync	Default NonSync	Default NonSync	
SidestepR	Default NonSync	Default NonSync	Default NonSync	Default NonSync	Default NonSync	Default NonSync	Default NonSy
Walk	Default NonSync	Default NonSync	Default NonSync	Cross Fade	Default NonSync	Default NonSync	Default NonSy

Visual Studio

NiActorManager – (also used by Animation Tool)

- •Handles loading and linking of KFM and NIF files
  - -can override, swap, or add sequences
- •SetTargetAnimation(SequenceID)
  - -based on keypresses, character velocity, sequence text tags, etc.
- •GetNIFRoot()
  - -add as child to main scenegraph
  - -update world position based on camera, etc.

Gameplay

- Most of the work was learning and adapting Gamebryo.
- Several custom classes adapted to non-uniform gravity
  - Camera, Controls, gravity forces, projectiles

Camera

- When in open space, expect camera to function like a spacecraft, with no set up vector… Yaw, Pitch, and Roll
- When on the ground, expect the camera's up vector to be perpendicular to the ground… Yaw and Pitch
- Implemented both modes separately and interpolation methods to smoothly transition from one camera to the other

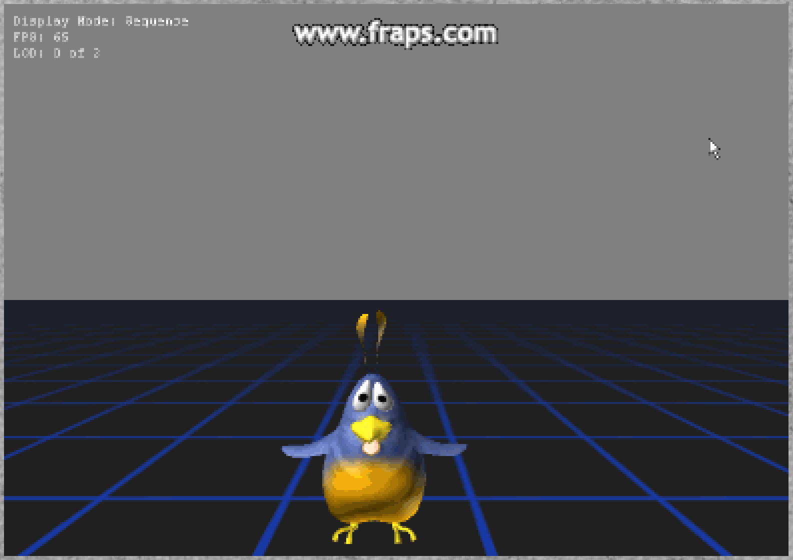
Gravity

- Completely physical gravity is nice
  - But it's not directable, and not as fun
- Implemented 2 types of gravity:
  - Static volumes that force actors inside their volume to feel only one gravity in a single direction. (gravity acceleration is constant vector)
  - Planet like gravity that attracts object flying through space and keeps actors from walking off the surface of the planet. (gravity acceleration falls off by square of distance)

Controls

- Similar controls to many FPS games
  - Some specific adaptions to make playing fun and easier to a broader range.
  - Controls play tested by novice and experienced user, feedback led to current control scheme

Demo!



```
Display Mode: Sequence
FPS: 65
LOD: 0 of 2
```

Tools & Languages

- Emergent Gamebryo 2.5 Evaluation
- NVIDIA/AEGIS PhysX
- Microsoft Visual Studio, C++
- Autodesk Maya 2008
- Nima PhysX Maya Plug-in

Functional Breakdown

- Our Work:
  - Animated Character
  - Camera
  - Controls
  - Projectile
  - Gravity
  - Explosion forces
  - Level geometry

- Gamebryo:
  - Rendering
  - Loading
  - Framework
  - Animation controler
- PhysX
  - Collision
  - Velocity integration

Contributions

- A novel game that combines a highly entertaining gameplay concept with a challenging jumping mechanic to allow mankind to gain an intuitive feeling for orbital physics.
- Good times.

Future Directions

- Learn, learn, learn some more
- Debug KFM import failures
- Integrate layered sequence animations
- Use text tags and timing information to improve transitions
- Implement obstacle class that a flying character would slide across
- Create more levels, possibly level editor
- Beta test using wider audience ranging from ‘|337’ rocket jumpers to complete novices
- Improve and upgrade gun & projectile system