Physics Meets Animation
Character Stunts in Just Cause 2

John Fuller – System Architect, Physics
Andreas Nilsson – Lead Gameplay Programmer
Talk Overview

Motion Control

Animation + Physics +IK

Parametric Animation

Effectors / Manipulators
Just Cause 2: Requirements

• Huge open world
• Fast-paced, over-the-top action
• Reactive environment
• High level of responsiveness
• Large number of game mechanics
• Large number of vehicles

Freedom!
Concept Video

videomatic_060918_01_xvid.av
Approach

- Small animation budget
- Large feature set
- Small animation staff budget

= Tiny

Procedural animation?
What It Is Not

• AI – Animation interface
• Path Planning
• Dynamic Motion Synthesis
• Procedural Controllers / Robotics

Motion Control
Motion Control

• Started with badly structured character control system
• Slow and cumbersome to create behaviors
• First: decoupled root motion from posture update
• Refactored functional elements into ‘Motion States’
Fast-paced Motion Transitions
Motion States - Root Node Update

- Desired motion:
  - Procedurally driven motion
  - Animation driven motion
  - Attached motion
- External influences:
  - Collision response
  - Gravity
Attached Motion

- Attached characters live in parent’s local space
- Character movement changes relation between parent and child
- Animated root node translation and rotation affects offset
Scaling for fixup / alignment

- Scale motion to realign for specific targets
- Introduces two constraints on the assets
  - Low curvature within the translation
  - No translation during contact with parent
- Animator has control over timing and acceleration
Rigid Body Proxy

• Physical effects applied in a controllable way
• Applying impulses to a ragdoll: less controllable
• Single rigid body represents entire character
• Can be constrained to other objects
Recoil
Custom Transition States

• Some transitions needed special care
• Transition states bridge between motion states with different velocities
• Applies custom velocities and impulses
• Crucial to fluid gameplay
• These are context dependent
What It Is Not

- AI
- Animation interface
- Path Planning
- Dynamic Motion Synthesis
- Procedural Controllers / Robotics

Animation / Physics / IK
Pre-visualization
Ragdoll / Animation / IK Blending

Systems influencing pose:

• Ragdoll
• Cling animations
• Hand and foot IK
• Aim Constraints
Control Flow

- Sample Animation Pose
- Foot / Hand IK attachment
- Update ragdoll
- Physics Update
- Aim constraints
- Skinning
Pose Driving

- Drive ragdoll towards animation pose (using impulses / joint motors)
- Not a keyframed ragdoll - can still respond to collisions
Transition from Ragdoll to Animated

1. Below a certain velocity, transition to Pose Matching state

2. Compare orientation with a number of Get-Up start frames

3. Drive ragdoll towards the closest start frame

4. When close to target pose, start the animation and blend to it
Spinning Ragdolls

Needed Over-the-top, extreme reactions to explosions

1. Applied impulses to torso and hips
   - Very ragdolly 😊
   - Occasional instability (stretching)

2. Evenly spread impulses on all bones
   - Lots of translation, not much spin
   - Synchronized swimmers (in-sync flailing animation)
   - Reminiscent of sprites!
Spinning Ragdolls

3. Vector field
   -> Get an axis perpendicular to explosion
   -> Evenly spread impulses to achieve rotation
   -> Still have synchronized swimmers

4. Randomness
   -> Vary the axis within a 45 degree cone

Note: Also drive towards flail animation
Authoring Ragdoll / Character setup

AFSM

S1

S2

S3

S4

Character Config

Ragdoll Files
Authoring Ragdoll / Character setup

```xml
<object name='Setting_RagdollVehicle'>
  <value name='name' type='string'>RagdollVehicle</value>
  <value name='ragdoll_file_id' type='string'>ragdoll_file_1</value>
</object>

<object name='parameters'>
  <value name='controller_type' type='string'>RIGID_BODY</value>
  <value name='map_anim_to_physics' type='int'>1</value>
  <value name='map_physics_to_anim' type='int'>1</value>
  <value name='blend_speed' type='float'>2</value>
  <value name='blend_weight' type='float'>0</value>
  <value name='tau' type='float'>0.95</value>
  <value name='damping' type='float'>0.35</value>
  <value name='proportional_recovery_velocity' type='float'>10.0</value>
  <value name='constant_recovery_velocity' type='float'>4.0</value>
  <value name='max_force' type='float'>10000.0</value>
  <value name='hierarchy_gain' type='float'>0.01</value>
  <value name='velocity_damping' type='float'>0.0</value>
  <value name='acceleration_gain' type='float'>0.35</value>
  <value name='velocity_gain' type='float'>0.35</value>
  <value name='position_gain' type='float'>0.35</value>
  <value name='position_max_linear_velocity' type='float'>1000.0</value>
  <value name='position_max_angular_velocity' type='float'>1000.0</value>
  <value name='snap_gain' type='float'>0.25</value>
  <value name='snap_max_linear_velocity' type='float'>0.1</value>
  <value name='snap_max_angular_velocity' type='float'>0.1</value>
  <value name='snap_max_linear_distance' type='float'>0.01</value>
  <value name='snap_max_angular_distance' type='float'>0.01</value>
</object>

<object name='keyframed_parts'>
  <value name='part_1' type='string'>Ragdoll_LeftHand</value>
  <value name='part_3' type='string'>Ragdoll_LeftFoot</value>
  <value name='part_4' type='string'>Ragdoll_RightFoot</value>
</object>
</object>
```
Authoring Ragdoll / Character setup

Anecdote:
• Rico freefall colliding state had artifacts
• Technical Animator diagnosed problem:
  • conflicting animation and constraints
• Tweaked ragdoll constraint limits
• Created a new Character Configuration
• Changed ‘Falling’ state to point to this character configuration
• Rico’s death sequence reworked in one morning, no coders involved
Physics Driven Animation
Ragdoll pros

• Feeling of presence
• Collision handling

Ragdoll cons

• Feeling of intention and awareness
• Poor momentum transfer
Traditional Link Between Animation and Physics

• Physics Event -> Animation Transition
• Results in:
  • Series of discrete animation states and transitions
  • Recognizable state machine style
  • Repetitive timing and movement patterns
• Artifacts typically combated with:
  • Shorter animations, more transitions, more complex trees?

But ... neither physics nor character behavior is discrete!!!
Parametric Animations

- Commonly used for navigation on ground?
- Smooth dynamic motion
Physics Driven Animation

- Parameterize blend nodes with parent’s motion
- Feed in continuous values to act as inputs to single states
- Result? Non-repetitive, smooth motion
Ragdoll Only
How does it work?

• All poses are baked into two animations
  • Upper row from left to right
  • Lower row from left to right

• Middle row is the result of blending

• Project parent’s angular velocity onto..
  • X-axis to determine blend weight
  • Y-axis to determine sample time
Multiple parameterizations create variation

Parachuting has the following inputs:
  - Acceleration, velocity and gamepad input

Riding motorcycle has the following inputs:
  - Suspension length rate of change
  - Speed
  - Orientation
  - Gamepad input
Physics Driven Animation
What It Is Not

• Animation interface
• Path Planning
• Dynamic Motion Synthesis
• Procedural Controllers / Robotics

Effectors / Manipulators
Animation Driven Impulses

• Wanted data driven physical effectors

• Animations contain annotations, e.g:
  - DOWNWARD-IMPULSE-LIGHT
  - DOWNWARD-IMPULSE-HEAVY

• Impulses applied to parent or target body

• E.g. foot down event, enter vehicle, some cling positions
Motorbike Tilt

- Let the player feel in control of the driver

- Forward-back controls player lean ... C.O.M. shift

- Re-align constraint limits on front and back

- Makes it easier to tip backwards

- Also allows for leaning forwards / backwards in air
The Almighty Grapple

- Physical constraint

- Can ‘tie’ nearly any two physics objects together

- Custom impulses applied: e.g. yanking, wall tether, dual tether two enemies, etc.

- Shorten the constraint to draw things together
Animation Driven Impulses
Findings
Problems we faced / Tips

Ragdoll Stability:

- Requires constant maintenance
- Animation poses must not violate constraint limits
  - Use different ragdolls to suit the context
- QA unfamiliar with problem domain
- Monitor edge cases: have a fallback
Problems we faced / Tips

Ragdoll Driving:
  • Varied quality at different speeds
  • Tried varying driving params with speed, ran out of time

Motion:
  • Transitions between Motion States took a lot of work
Problems we faced / Tips

Blending:
- Noisy physics signal - filter
- Blending away from a parametric blend node can be difficult

Dependencies:
- Difficult to tweak without side effects
Important decisions we made

• We separated motion state from pose generation

• Many states had different control flow for IK / Animation / Physics.
  • We were able to vary this control flow for each state.
  • Not quite a dynamic ‘shader pipeline’, but flexible

• Exposed elements of the character configuration to content creators
Advantages of using Physics

• Cheap variation - few added animations

• Rich context data to drive animation blending

• Collision response enriches feeling of interaction and presence

• Fun emergent gameplay, e.g. grapple
Disadvantages of using Physics

- Requires constant maintenance and tuning
- Hard to preview final visual outcome
- Requires expertise across the organization, e.g. game designers, animators, QA
Thanks!

Just Cause 2 Team
Avalanche Studios

Eidos
Square Enix
Havok
Q & A