

Physical Gameplay in Half-Life 2

presented by
Jay Stelly

Physical Gameplay in Half-Life 2

- ⊕ New technology that hadn't been successfully integrated into our genre
- ⊕ Technical solutions not very well understood
- ⊕ Obvious visual payoff
- ⊕ Opportunity was to integrate with gameplay
- ⊕ Both a game design problem and a technical problem

High-level strategy

- ④ Don't build the simulator
- ④ Don't add features to the simulator (until it becomes necessary)
- ④ Differentiate the product by depth of gameplay integration, not incremental simulator features or quality
- ④ Engineer tools and solutions in the game design space

Half-Life 2 Timeline for Physics

- ⊕ Inspired by physics demos
- ⊕ Generated a bunch of ideas
- ⊕ Licensed physics simulator
- ⊕ Took some time for game designers to really internalize physics technology
 - Built a bunch of prototypes
 - Built a bunch of design tools & logic

Half-Life 2 Timeline for Physics (continued)

- ④ Gameplay mechanics experiments
- ④ Solved some technical problems
- ④ Cut & focus pass
- ④ Solved more technical problems
- ④ Incrementally delivered a stable system
 - Valuable features at each deliverable
- ④ Polished and shipped the game

Physics prototypes (pre-production)

- ⊕ Zombie basketball
- ⊕ Watermelon skeet shooting
- ⊕ Glue gun
- ⊕ Danger Ted playset
- ⊕ Toilet crossing

Cut & Focus pass

- ④ How can we tell which gameplay idea is better?
- ④ How many gameplay ideas do we need?
- ④ How can we measure or change the difficulty of this gameplay?
- ④ How are we going to turn these prototypes into shippable gameplay?

Are there metrics or analyses that will lead to better gameplay?

Is there a systematic way to move these ideas forward?

What are the technical problems we'll need to solve?

Game design

- ⊕ Game design can be reduced to training and testing:
- ⊕ A game design is a set of player experiences that:
 - trains a player with specific skills and knowledge
 - allows or requires the player to demonstrate that skill or knowledge
 - is presented with style.

Game design is engineering (at least a bunch of it is)

- ⊕ Define success
- ⊕ Identify constraints
- ⊕ Generate ideas
- ⊕ Analyze solutions
- ⊕ Build prototypes
- ⊕ Test results
- ⊕ Measure success
- ⊕ Re-examine constraints

Engineering training and testing

- ④ Measurable criteria
- ④ Models & Analysis
 - Cost / benefit
- ④ Tradeoffs
- ④ How to cut
- ④ How to compare
- ④ How to solve backwards for requirements
- ④ How to measure value

Soundscape: d1_trainstation.Interrogation
Soundscape: d1_trainstation.Turnstyle
SetPoliceGoal: npc_metropolice (cupcop) unable to find ai_goal_police:



Tools for training

- ⊕ By example
- ⊕ Clues then deduction
- ⊕ Cliché
- ⊕ Explicit test (assertion)
- ⊕ Sandbox / toy / experiment
- ⊕ Practice
- ⊕ Forced choices

Obstacles to training

- ⊕ Combat
- ⊕ Peril
- ⊕ Basically anything that forces the player to make decisions
- ⊕ Reactions – rely on past skills & knowledge

Improving training

- ⊕ Make it clear that it's ok to experiment or fail
- ⊕ Sell forced choices with style
- ⊕ Suggest experiments
- ⊕ Story is not an obstacle to training



© 2007 Valve Corporation. All Rights Reserved.



© 2007 Valve Corporation. All Rights Reserved.

Player value as a metric for skills and knowledge

- ④ Each piece of skill or knowledge must have value or get cut from your game
- ④ There is a limit to the total number of things you can train in a game
- ④ Having a skill or piece of knowledge interact with another increases the value of both
- ④ Requiring a piece of skill or knowledge to pass a test increases its value to the player
- ④ These relationships form an economy that can be analyzed and optimized
- ④ At Valve we call this “design economy.”

Constraints from Half-Life

- ⊕ Breakable objects – crowbar
- ⊕ Physics needs to interact with core combat gameplay
 - Collisions that cause damage
 - Players and NPCs use physics as cover
- ⊕ Physics needs to extend core puzzle gameplay

Integrating physics with Half-Life is difficult

- ⊕ Physics is reasonably intuitive, but doesn't "just work" for a bunch of reasons.
- ⊕ Most game designers don't completely understand the physics simulation technology, implementing their designs makes understanding the simulator really important.
- ⊕ Game logic may place impossible requirements on a physics simulation – requiring code to be written that straddles the boundary between game design and physics technology.

Design interface

- ⊕ Educating designers in physics
- ⊕ Decomposing machines into physics blocks
- ⊕ Unfamiliar units (e.g. torque, impulses)
- ⊕ Tuning parameters
- ⊕ Complex sets of variables imply calculations
 - I want this part of this machine to spin at this speed
 - I want this plank to be stable enough to support the player, but only until he reaches this point
- ⊕ Deliver technology incrementally
 - Only a few features to learn at a time
- ⊕ Need a physics expert to support designers

Latency & Continuity

- ④ Most physics engines interact with the game in discrete steps of time
- ④ Changes to the state of the system are often queued until the next update/step
- ④ Game rules are often discontinuities in state
 - I want to break this object on collision
 - You can only break objects at time steps
 - Collisions occur between time steps
 - Built support for this by resetting in the future
- ④ Run until the next collision is ideal, but not practical

Speculation

- ⊕ Reserving space (Inventory, creating objects)
- ⊕ Motion planning
- ⊕ Collision detection without physics (tools, queries)
 - Built tools and query layer
 - Critical problem for our AI system
 - Built in-house speculative collision solver

Overdetermined systems

- ⊕ simulation variables
- ⊕ design variables
- ⊕ design criteria
 - gravity gun movement vs. damage
 - zombie car trap
- ⊕ Superman problem

Simulation failure

- ⊕ Objects stuck in each other
- ⊕ Not settling
- ⊕ Valid for physics invalid for game design
- ⊕ Simulator explodes
- ⊕ Game design constraints that can't be satisfied
- ⊕ Create objects in solid space

Conclusions

- ④ Engineer your gameplay mechanics
- ④ Use analysis and design economy to intentionally improve your game design
- ④ Many technical problems remain with integrating physics. You can solve some of these with design constraints, but plan to invest in technology.
- ④ Plan for failure cases and be sure to ask, “is this failing as a result of desirable gameplay?”