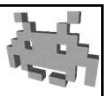
3D video games

Collision Handling

Marco Tarini

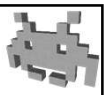
1

Course Plan



lec. 1: Introduction
lec. 2: Mathematics for 3D Games
lec. 3: Scene Graph
lec. 4: Game 3D Physics +
lec. 5: Game Particle Systems
lec. 6: Game 3D Models
lec. 7: Game Textures
lec. 8: Game 3D Animations
lec. 9: Game 3D Audio
lec. 10: Networking for 3D Games
lec. 11: Artificial Intelligence for 3D Games
lec. 12: Game 3D Rendering Techniques

3



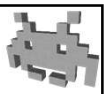
Collision Handling: a preliminary optimization

- Two types of objects in a game physics:
  - static objects
    - Never move (speed = 0)
    - Part of setting, background
    - Can affect other objects, not affected by other objects
  - non-static objects
    - Can move around (for any reason)

	Static	Movable
Static	⊘	One Way
Movable	One Way	Two Ways

- Two types of collisions:
  - one-way : a non-static object with a static object
  - two-ways : a non-static object with a non-static object

4

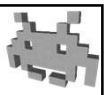


Collision Handling

- Collision detection
  - find out when they occur
- Collision response
  - compute their effects

5

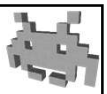
Collision response



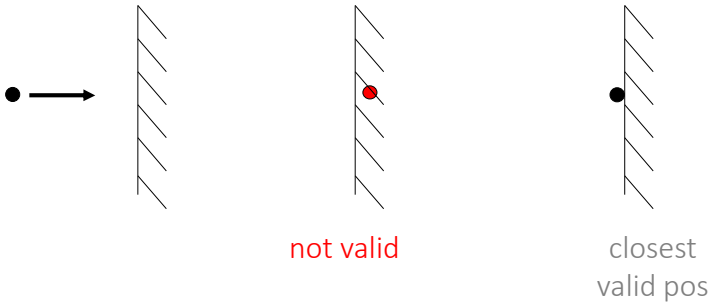
- Enforce non-penetration
  - place objects in valid positions
  - (*when to:* **always**)
- Impacts
  - add impulses (rebounces, etc)
  - (*when to:* collision occurred now, but not in the pref frame)
- Frictions between the two objects
  - energy dissipation
  - (*when to*: from 2° consecutive step of collision)
- Ad-hoc effects
  - breaking of objects, gameplat effects (HP loss?), etc (scripts)
  - (*when to - if at all:* entirely gameplay dependent)

6

Enforcing non-penetration

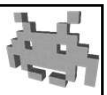


- Invalid position?
  - strategy 1: revert to last valid pos (easy to do, not ideal)
  - strategy 2: project to closest valid pos (necessary, in PBD)



7

Enforcing non-penetration



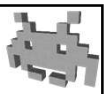
- With Position Based Dynamics:
  just another positional constraint
  - bonus: velocity updates
    (similar to inelastic impacts)
  - but we will need to explicitly compute
    impacts if we want a better control
    of the behavior
- How to enforce this constraint:
  - *one-way* :
    only displace the movable objects
    by the minimal amount
  - *two-ways* :
    move them both,
    minimizing the summed squared displacements × the mass

Note: anymmetrical constraint (> not =)

A big practical problem ☹ :
The presence of the constraint it is not known a-priori.

8

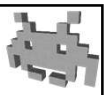
Frictions



- Apply on prolonged contact
  - collision with an object that was colliding last frame too
- How to:
  - Apply forces with
    direction opposite to current velocity, and
    magnitude proportional to speed
  - or (more simply): velocity damping

9

Resolving the impacts



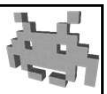
so, with an **impulse**

- **Sudden** velocity change
  - resolve the impact = determine the new velocities $\vec{v}_{new}$
  - equivalently, determine the impulses $\vec{\imath} = (\vec{v}_{new} - \vec{v}_{old}) \cdot m$
- *All* impacts preserve total **momentum** $m \cdot \vec{v}$
  - No matter what
- To resolve the impacts, we need further assumptions, different for each type of impact…
  - **elastic** impact
  - **inelastic** impact
  - or anything in between

a vector
(ita: *«quantità di moto»*)

10

Different type of impacts

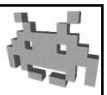


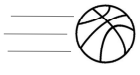
- (completely) **elastic** impact

- (completely) **inelastic** impact

11

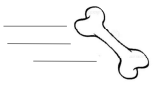
“Bounciness” (or impact elasticity)





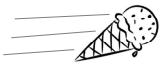
“Bounciness” = 1.0

…



“Bounciness” = 0.5

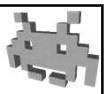
…



“Bounciness” = 0.0

12

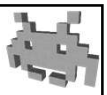
“Bounciness” (or impact elasticity)



- Elastic impact: no energy lost
- Inelastic impact: energy losses
  - e.g. objects are damaged, heat is produced…
- “Bounciness”: a (made up) property of physical objects in games
  - It models the behavior of the object under impacts, as a mix between the two extreme behavior above
  - Associated by designers to all virtual objects in the game
- Note: nothing of this is how stuff really works!
  - not even for the two extremes
  - It’s a cheap approximation (especially for mixed bounciness)
  - Remember: we are just shooting for plausibility

13

What about this impact?

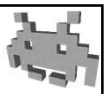


"Bounciness" = ???

- Practical solution:
  adopt some formula between the
  bounciness values associated to the two objects
  - For example: avg, min, max
  - It's a choice of the game engine
  - (can be hard-wired in the physics engine,
    or exposed to the users)

14

Assumptions for different types of impact



- (completely) elastic impact
  - preservation of total kinetic energy $\frac{1}{2}m \cdot \|\vec{v}\|^2$ (a scalar)
  - impulse direction = normal of impacted point
- (completely) inelastic impact
  - After the impact, the two bodies have the same velocity
  - (the impact momentarily "glued them together")
- Mixed cases:
  - Solve for both cases, then interpolate results
  - Interpolation weight is "bounciness"

  (a brutal, practical solution – not correct but plausible)

15