Introduction to Chipmunk

Marco Bancale & Sveinn Fannar Kristjánsson

What's Chipmunk?

CHIPMUNK

Game Dynamics

What's Chipmunk?

- 2D physics engine written in C
- Cross platform
- Two main editions:
  - Chipmunk (vanilla C version, free)
  - Chipmunk Pro (C, Obj-C, more features, not free)
- Hundreds of games shipped
- Written by Scott Lembcke, a very smart guy

We will use Chipmunk Pro Trial in this course.

Lots of cool features!

- Sleeping objects
- Fast collision detection
- Fast impulse solver
- Flexible collision filtering system
- Circle, convex polygon, beveled line segment collision primitives
- Many language bindings (Obj-C, Python, Ruby, C++, ...)
- Lightweight and no dependencies
- Raycasting
- Well documented
- Optimized for mobile devices
- Joints

Chipmunk Basics

Four basic object types:

- Rigid bodies
  - Hold physical properties (mass, position, rotation, etc...)
  - Don’t have a shape until one is attached to them
  - Better if 1:1 pixel/unit correlation with sprites
- Collision shapes
  - Circles, polygons, lines
  - One body can have many shapes
  - Hold surface properties (friction, elasticity, etc...)
- Constraints/joints
- Spaces
  - Containers for simulated objects
  - Bodies, shapes and joints must be added to a space
  - Control the whole simulation

Rigid bodies

ChipmunkBody / cpBody

- Static and non-static
- Properties:
  - Mass
  - Moment of inertia
  - Center of gravity
  - Linear velocity
  - Applied force
  - Rotation
  - Angular velocity
  - Applied torque
  - Data pointer
- Forces and impulses
- Bodies can sleep
- Don’t modify a body’s position directly!

Shapes

ChipmunkShape / cpShape

- Three types:
  - Circle (fastest and simplest)
  - Line segment (mainly for static bodies)
  - Convex polygon (slowest, most flexible)
- Properties:
  - Body it is attached to
  - Space it is contained in
  - Sensor
  - Elasticity
  - Friction
  - Group
  - Layer
  - Data pointer
- Need to be attached to a body
- Need to be added to a space
- Shapes on the same body don’t generate collisions

Spaces

ChipmunkSpace / cpSpace

- Must be “stepped”
- Always step by a CONSTANT DELTA
- Accuracy can be tweaked by setting the number of iterations
- Properties:
  - Iterations
  - Global gravity
  - Global damping
  - Data pointer
  - Other advanced properties
- Offers API to add/remove bodies/shapes/constraints
- Uses delegation pattern for handling collisions

Collision detection

- Collisions are detected and resolved during a step
- Application provides collision handlers
- Four handlers:
  - “begin” - called when two shapes just started touching for the first time in this step; collision can be marked as ignored by returning false
  - “preSolve” - called while two shapes are touching; collision values can be modified on the fly
  - “postSolve” - called after a collision response has been processed; good to retrieve resulting forces
  - “separate” - called when two shapes just stopped touching

Tying it all together

- Use a 1:1 conversion between pixels and Chipmunk's units
- Step with a constant delta; step multiple times within a single frame if the frame delta is too big
- Use the CCPhysicsSprite class from Cocos2D for physics based sprites
- Be careful when removing bodies and shapes from a space:
  - Don't do it in a collision handler
  - Always remove a body and its shapes at the same time

Autogeometry

- Fantastic feature only available in Chipmunk Pro
- Extracts collision shapes from images (!)
- Extremely fast, it can be done every frame to create deformable/ destructable terrain
- Image data can come from a file or in-memory texture

Part 1

GOAL:
Make the mountain and the tanks part of the physics simulation

Steps needed:

- Initialize Chipmunk
- Use the Autogeometry feature on the mountain
- Add an invisible floor
- Refactor the Tank class to use Chipmunk
- Implement the "update" method

Part 2

GOAL:
The tanks can shoot!

Steps needed:

- Design and create a Projectile class
- Make the Tank shoot a Projectile
- Implement collision handlers