PhysPix: Instantaneous Rigid Body Simulation of Rasters

Domagoj Baričević, James Schaffer, Theodore Kim
University of California, Santa Barbara*

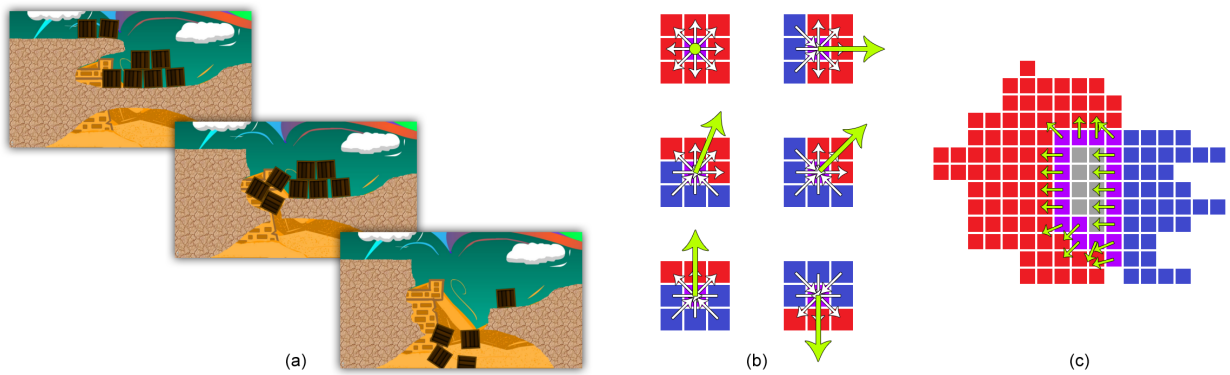

Figure 1: *From left to right: a) Terrain destroyed by erasing the ground bitmap, the rigid-body simulation is instantly updated. b) Some example pixel normals - red pixels belong to the reference object, blue pixels belong to the colliding object, the purple center pixel represents the actual intersection, white arrows show partial kernel normals, and the local pixel normal is shown in green. c) Two intersecting objects (red and blue) and their intersection - critical pixels are highlighted in purple while gray pixels are not used to determine normal; local normals for each pixel are shown in green, these are averaged to generate the surface normal for the collision.*

1 Introduction

Modern physics engines process collisions by leveraging vector representations (e.g. *Box2D* or *Open Dynamics Engine (ODE)*), which means that artists who work with pixel-based 2D content must map their pixel drawings onto representations such as Delaunay triangulations [Shewchuk 1996]. Effects such as destruction then require remeshing, which can be onerous to perform at runtime. The alternative is pixel-perfect collision handling, but past games such as *Worms!* and *Scorched Earth* that use this approach have not attempted true rigid body simulations. We present *PhysPix*, a 2D rigid body simulation framework based on pixels. PhysPix allows 1) artist control over the exact boundaries used for objects in the simulation, 2) natural bitmap-based support for destruction, and 3) an intuitive painting interface for properties such as non-uniform weight distributions.

2 Approach

PhysPix asks artists for a single bitmap for each object, and uses this bitmap 'as-is' in the rigid body simulation. The challenge lies in extracting the components needed for rigid body response (surface normal, penetration depth) from arbitrary bitmaps. Sequential computation can be expensive, even for simple cases.

Weight distribution and collision geometry - information that determines the behavior of a rigid body - are naturally encoded into a bitmap for 2D objects. The minimum amount of information required from the artist is the sprite that represents the object graphically, but the artist can also additionally specify two optional bitmaps: the collision mask, and the weight distribution mask. By default PhysPix will use the alpha channel of the sprite to determine the collision shape of the object, the collision mask can be used to override this shape with a custom one. PhysPix computes the center of gravity and inertia tensor from the supplied bitmaps. By default this information is extracted from the alpha channel of the sprite (or the collision mask), but can be be optionally extracted from the weight distribution mask. PhysPix's default method treats each opaque pixel in the sprite as unit mass. If the weight distribution mask is specified, each pixel is assigned a relative mass based on the intensity of its color. Using the explicit method allows a content creator to intuitively specify the precise weight distribution of any object in any arbitrary way - a feature which is more difficult to expose in a vector-based engine.

In image processing, the task of summarizing bitmaps has been well studied and is known to easily map onto parallel processors. By processing bitmaps in parallel using compute shaders, PhysPix makes the dynamic extraction of normal and depth from intersecting bitmaps practical for real-time simulation. At runtime, PhysPix processes collisions between bitmaps on the GPU and uses a 2D-configured ODE for its rigid body simulation. At the time of collision, the intersecting bitmap (collision raster) is thrown onto the GPU to determine the collision's normal, depth, and approximate location. The location of the collision is simply the collision intersection's center of mass. For each pixel in the collision raster, a local normal is generated by inspecting a 3x3 kernel centered at the pixel (Fig 1b). Only the pixels on the surface of the collision (critical pixels) generate local normals that are useful. For each critical pixel, the local normal is generated by inspecting the pixel's 8 possible neighbors. This method gives a normal for each pixel on the surface of the collision (Fig 1c). Local normals are averaged into the global normal for the collision raster. The global normal is then leveraged to determine the depth of the collision: the collision raster is rotated so that the normal is aligned with the rows in the bitmap, then the GPU is used to find the maximum of these rows which corresponds to the depth of the collision.

PhysPix enables fully destructible environments on the individual pixel level. Since all collisions are represented as raster intersections, updating the collision geometry is a simple matter of updating the bitmap representing the object or environment (Fig 1a).

References

SHEWCHUK, J. R. 1996. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Applied computational geometry towards geometric engineering*. Springer, 203–222.

*e-mail: {domagoj, james_schaffer}@cs.ucsb.edu, kim@mat.ucsb.edu

SIGGRAPH 2013, July 21 – 25, 2013, Anaheim, California.

ACM 978-1-4503-2261-4/13/07