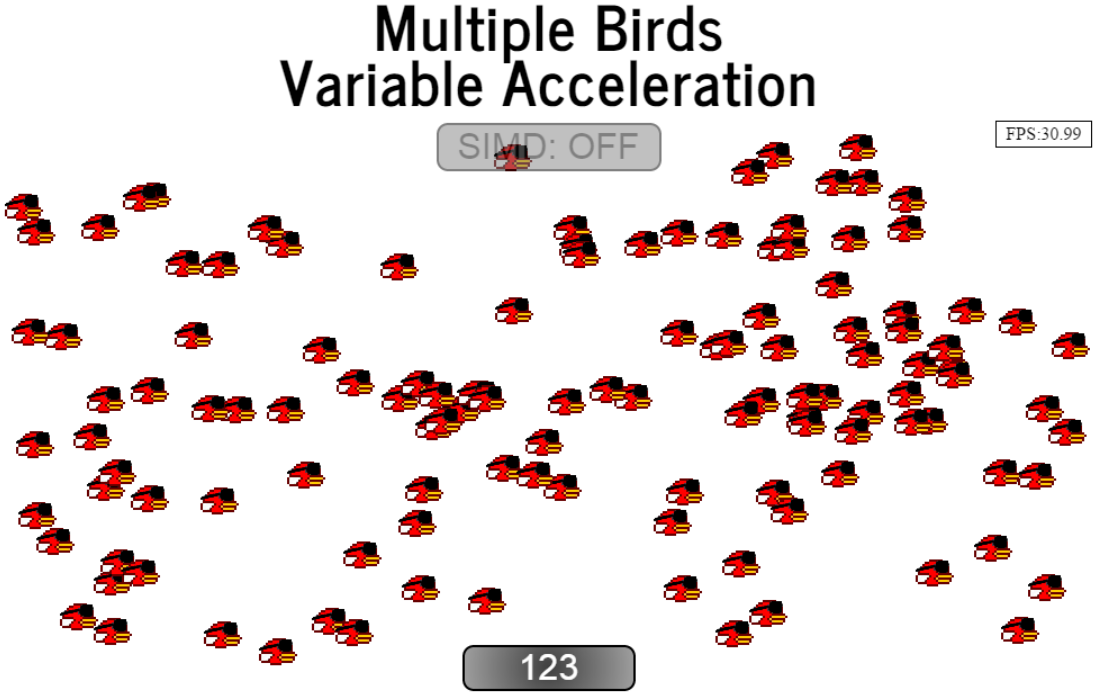
Box2D with SIMD in JavaScript

Ningxin Hu
Peter Jensen
Intel Corporation
October 20, 2014

‘SIMD Programming in JavaScript’



$$v_{n+1} = a\Delta t + v_n$$

$$s_{n+1} = \frac{1}{2}a(\Delta t)^2 + v_n\Delta t + s_n$$

‘SIMD Programming in JavaScript’

```
function updateAllSimd(timeDelta) {
  var steps        = accelData.steps;
  var accelCount   = accelData.values.length;
  var subTimeDelta = timeDelta/steps/1000.0;

  var posArrayx4            = new Float32x4Array(posArray.buffer);
  var velArrayx4            = new Float32x4Array(velArray.buffer);
  var maxPosx4              = SIMD.float32x4.splat(maxPos);
  var subTimeDeltax4        = SIMD.float32x4.splat(subTimeDelta);
  var subTimeDeltaSquaredx4 = SIMD.float32x4.mul(subTimeDeltax4, subTimeDeltax4);
  var point5x4              = SIMD.float32x4.splat(0.5);

for (var i = 0, len = (actualBirds+3)>>2; i < len; ++i) {
    var newVelTruex4;
    var accelIndex = 0;
    var newPosx4   = posArrayx4.getAt(i);
    var newVelx4   = velArrayx4.getAt(i);
    for (var a = 0; a < steps; ++a) {
      var accel   = accelData.values[accelIndex];
      var accelx4 = SIMD.float32x4.splat(accel);
      accelIndex  = (accelIndex + 1) % accelCount;
      var posDeltax4;
      posDeltax4   = SIMD.float32x4.mul(point5x4, SIMD.float32x4.mul(accelx4, subTimeDeltaSquaredx4));
      posDeltax4   = SIMD.float32x4.add(posDeltax4, SIMD.float32x4.mul(newVelx4,subTimeDeltax4));
      newPosx4     = SIMD.float32x4.add(newPosx4, posDeltax4);
      newVelx4     = SIMD.float32x4.add(newVelx4, SIMD.float32x4.mul(accelx4, subTimeDeltax4));
      var cmpx4    = SIMD.float32x4.greaterThan(newPosx4, maxPosx4);
      newVelTruex4 = SIMD.float32x4.neg(newVelx4);
      newVelx4     = SIMD.int32x4.select(cmpx4, newVelTruex4, newVelx4);
    }
    posArrayx4.setAt(i, newPosx4);
    velArrayx4.setAt(i, newVelx4);
  }
}
```

- Nice ~3x speedup!
- Only One dimensional – Box1D
- Only one body shape
- No body->body collision detection
- No rotation
- No rotation velocity

Is this applicable to a real physics engine like Box2D?

Agenda

Box2D

- Background
- Uses
- Basics
- Implementations (native and JS)

SIMD in JavaScript

- Basics
- Availability

Agenda

Emscripten

- Basics. How does it work?
- Native SIMD -> JavaScript SIMD
- JavaScript Bindings

Box2D SIMD opportunities

- Performance profiles
- Vector/matrix math
- Constraint solvers (position, velocity, time-of-impact)

Summary

- What worked and what didn't

Box2D Background

- Written by **Erin Catto**
- Written in C++
- First released as "Box2D Lite", a demonstration engine to accompany a physics presentation given by Erin Catto at GDC 2006.
- Released as open source on Sourceforge on September 11, 2007
- Version 2.0 launched on March 2008, introducing continuous collision detection and a revamped API.
- The latest version is v2.3.1
- Hosted here:
  - box2d.org

Box2D Uses

Games

- Crayon Physics Deluxe, Limbo, Rolando, Fantastic Contraption, Incredibots, Angry Birds, Tiny Wings, Transformice, Happy Wheels, ...

Game Engines

- Unity2D, Construct2, Cocos2D, Ludei, Corona, libGDX, Godot

Other Uses

- LiquidFun

Box2D Basics

Simulates a 2D world with interacting rigid bodies of various shapes

- Create the world

```
b2Vec2  gravity(0.0f, -10.0f);
b2World world(gravity);
```

- Add the bodies

```
b2BodyDef bd;
bd.type       = b2_dynamicBody;
bd.position   = b2Vec2(-7.0f, 0.75f);
b2Body *body = world.CreateBody(&bd);
```

Box2D Basics

- Add Fixtures to the bodies. Note: A body can have multiple fixtures

```
b2CircleShape shape;
shape.m_radius = 2.0f;
body->CreateFixture(&shape, 5.0f); // 5.0f is density
```

- Set the world in motion

```
world.Step(1.0f/60.0f, 3, 3);
// 1. param: time delta
// 2. param: velocityIterations
// 3. param: positionIterations
```

Box2D Implementations

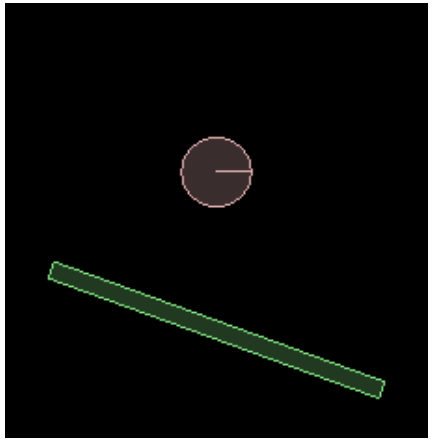
Box2D has been ported to many different languages

- Flash:
  - http://www.box2dflash.org/
- Java:
  - http://www.jbox2d.org/
- Python:
  - http://code.google.com/p/pybox2d/
- C#:
  - http://code.google.com/p/box2dx/
- **Javascript:**
  - Port of box2dFlash
  - http://code.google.com/p/box2dweb/
- **JavaScript (asm.js):**
  - Automatic build using Emscripten (by Alon Zakai)
  - https://github.com/kripken/box2d.js/

Box2D Using C++

C++

```
b2BodyDef bd;

// Create a stick body
bd.type = b2_staticBody;
bd.position.Set(0.0f, 2.0f);
b2Body *stick = m_world->CreateBody(&bd);

// Attach a rectangle fixture to the stick
b2PolygonShape rect;
rect.SetAsBox(0.5f, 10.0f, b2Vec2(0.0f, 0.0f),
              70.0f * b2_pi/180.0f);
stick->CreateFixture(&rect, 0.0f);

// Create a ball body
bd.type = b2_dynamicBody;
bd.position.Set(0.0f, 20.0f);
b2Body *ball = m_world->CreateBody(&bd);

// Attach a circle fixture to the ball
b2CircleShape circle;
circle.m_radius = 2.0f;
ball->CreateFixture(&circle, 5.0f);
```

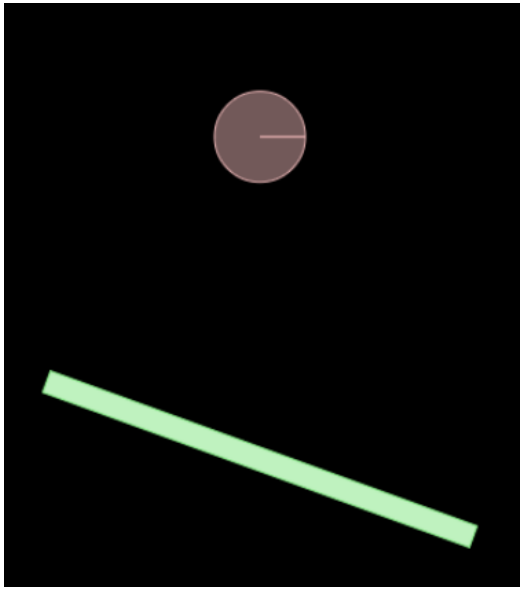


- A world consists of bodies
- Bodies have positions and types
- Bodies are composed of fixtures
- Fixtures have shapes and densities
- Body placement: center (x,y) and rotation (a)
- Body velocity: velocity of center (x,y) and angular speed (w)

Box2D Using Box2DWeb

Box2DWeb

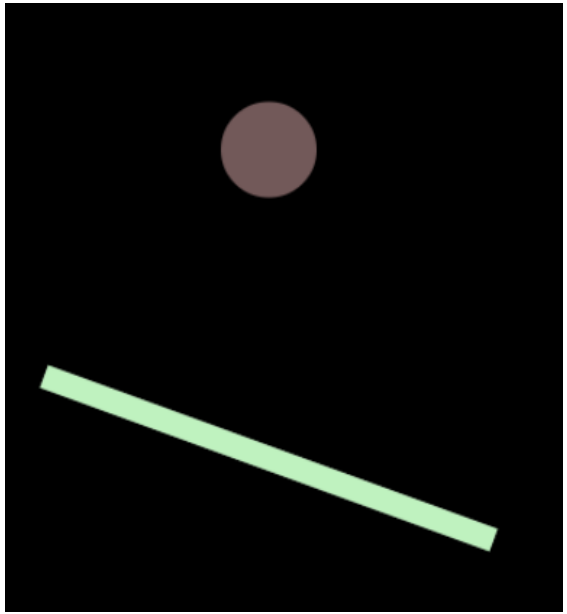
```
var bd = new b2BodyDef();

// Create a stick body
bd.type = b2Body.b2_staticBody;
bd.position.Set(0.0, 0.0);
var stick = world.CreateBody(bd);

// Attach a rectangle fixture to the stick
var rect = new b2PolygonShape();
rect.SetAsOrientedBox(0.5, 10.0, new b2Vec2(0.0, 0.0),
                      70.0 * Math.PI / 180.0);
stick.CreateFixture2(rect, 0.0);

// Create a ball body
bd.type = b2Body.b2_dynamicBody;
bd.position.Set(0.0, 20.0);
var ball = world.CreateBody(bd);

// Attach a circle fixture to the ball
var circle = new b2CircleShape();
circle.m_radius = 2.0;
ball.CreateFixture2(circle, 5.0);
```



Box2D using box2d.js (asm.js)

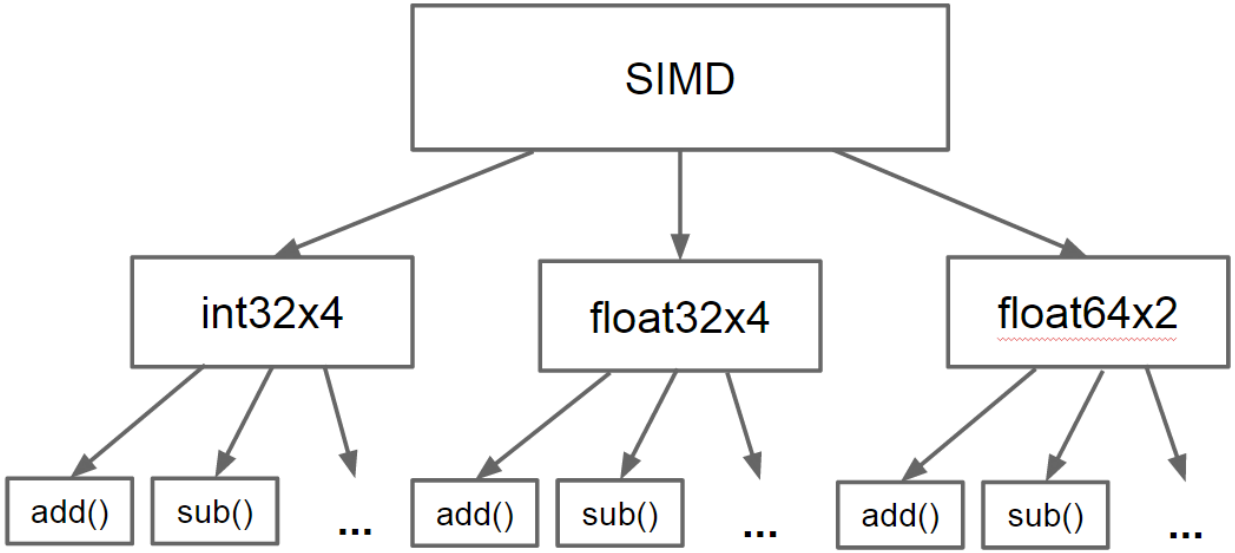
Box2d.js (asm.js)

```
var bd = new b2BodyDef();

// Create a stick body
bd.set_type(Box2D.b2_staticBody);
bd.set_position(new b2Vec2(0.0, 0.0));
var stick = world.CreateBody(bd);

// Attach a rectangle fixture to the stick
var rect = new b2PolygonShape();
rect.SetAsBox(0.5, 10.0, new b2Vec2(0.0, 0.0),
              70.0 * Math.PI / 180.0);
stick.CreateFixture(rect, 0.0);

// Create a ball body
bd.set_type(Box2D.b2_dynamicBody);
bd.set_position(new b2Vec2(0.0, 20.0));
var ball = world.CreateBody(bd);

// Attach a circle fixture to the ball
var circle = new b2CircleShape();
circle.set_m_radius(2.0);
ball.CreateFixture(circle, 5.0);
```



SIMD in JavaScript

Simple set of **types and primitives** that can be efficiently mapped to CPU instructions:



SIMD in JavaScript

SIMD in C/C++

```
float simdAverage(float *src, int len) {
  __m128 sumx4 = _mm_set_ps1(0.0f);

  for (int i = 0; i < len; i += 4) {
    sumx4 = _mm_add_ps(
              sumx4, _mm_loadu_ps(src));
    src += 4;
  }
  float sumx4_mem[4];
  _mm_storeu_ps(sumx4_mem, sumx4);
  return (sumx4_mem[0] + sumx4_mem[1] +
          sumx4_mem[2] + sumx4_mem[3])/len;
}
```

SIMD in JavaScript

```
function simdAverage(src, len) {
  var sumx4 = SIMD.float32x4.splat(0.0);
  var srcx4 = new Float32x4Array(src.buffer);
  for (var i = 0, n = len/4; i < n; ++i) {
    sumx4 = SIMD.float32x4.add(
              sumx4, srcx4.getAt(i));

  }


  return (sumx4.x + sumx4.y +
          sumx4.z + sumx4.w)/len;
}
```

SIMD offers a potential ~4x speedup

SIMD in JavaScript

- Active collaboration between Google, Mozilla, Microsoft, and Intel
- Spec/polyfill/benchmarks available here:
  - github.com/johnmccutchan/ecmascript_simd
- 1st stage approval for inclusion in ES7 by TC39
- Prototypes available for Firefox nightly and Chromium:
  - peterjensen.github.io/idf2014-simd

Emscripten

- Developed by Alon Zakai/Mozilla:
  - github.com/kripken/emscripten
- Compiles LLVM bitcode to JavaScript
- Supports most of _mm_X_ps() intrinsics

```
% emcc -O2 -g demo02.c
```

```
float simdAverage(float *src, int len) {
  __m128 sumx4 = _mm_set_ps1(0.0f);
  for (int i = 0; i < len; i += 4) {
    __m128 v = _mm_load_ps(src + i);
    sumx4 = _mm_add_ps(sumx4, v);
  }
  float sumx4_mem[4];
  _mm_store_ps(sumx4_mem, sumx4);
  return (sumx4_mem[0] + sumx4_mem[1] +
          sumx4_mem[2] + sumx4_mem[3])/len;
}
```

```
function _simdAverage($src, $len) {
 $src = $src | 0;
 $len = $len | 0;
 var $add$i         = SIMD_float32x4(0, 0, 0, 0),
     $i$011         = 0,
     $sumx4$0$lcssa = SIMD_float32x4(0, 0, 0, 0),
     $sumx4$010     = SIMD_float32x4(0, 0, 0, 0),
     sp = 0;
 sp = STACKTOP;
 if (($len | 0) > 0) {
  $i$011 = 0;
  $sumx4$010 = SIMD_float32x4_splat(Math_fround(0));
  while (1) {
   $add$i = SIMD_float32x4_add(
              $sumx4$010,
              SIMD_float32x4_load(
                buffer, $src + ($i$011 << 2) | 0));
   $i$011 = $i$011 + 4 | 0;
   if (($i$011 | 0) >= ($len | 0)) {
    $sumx4$0$lcssa = $add$i;
    break;
   } else $sumx4$010 = $add$i;
  }
 } else
 $sumx4$0$lcssa = SIMD_float32x4_splat(Math_fround(0));
 STACKTOP = sp;
 return +((+$sumx4$0$lcssa.w +
           (+$sumx4$0$lcssa.z +
           (+$sumx4$0$lcssa.x +
            +$sumx4$0$lcssa.y))) / +($len | 0));
}
```

Emscripten JavaScript -> C++ Bindings

User JavaScript Code

```
var bd = new b2BodyDef();
// Create a stick body
bd.set_type(b2_staticBody);
bd.set_position(new b2Vec2(0.0, 0.0));
```

box2d.idl

```
enum b2BodyType {
  "b2_staticBody",
  "b2_kinematicBody",
  "b2_dynamicBody"
};

interface b2BodyDef {
  void b2BodyDef();
  attribute b2BodyType type;
  attribute b2Vec2 position;
  …
}
```

Box2D C++ declarations

```
enum b2BodyType {
  b2_staticBody = 0,
  b2_kinematicBody,
  b2_dynamicBody
};
struct b2BodyDef {
  b2BodyDef();
  b2BodyType type;
  b2Vec2     position;
  …
}
```

Emscripten commands to tie it all together

```
### generate box2d_glue.js and box2d_glue.cpp
% python webidl_binder.py box2d.idl box2d_glue
### generate box2d.js
% em++ box2d.bc box2d_glue.cpp \
  --post-js box2d_glue.js \
  -o box2d.js
```

Box2D SIMD Opportunities

No publicly available use of SIMD in Box2D

- At least we couldn't find any
- We knew it was going to be challenging

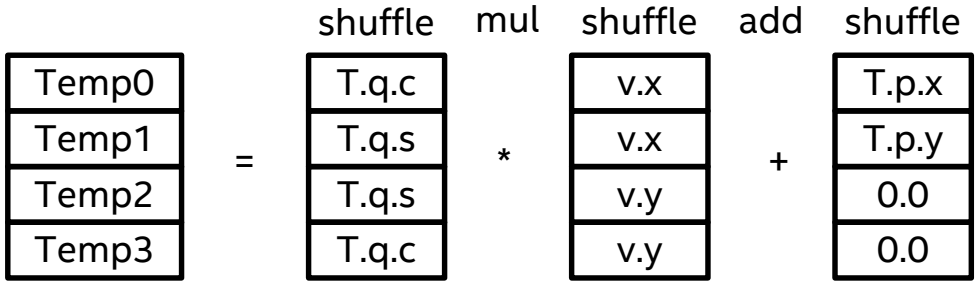
How to find opportunities

- Use a good performance profiler
- Low Level approach:
  - Look for sequences of arithmetic operations that can be combined
- High Level approach:
  - Look for loops where iteration count can be /4

Box2D SIMD Opportunities – Low Level

```
inline b2Vec2 b2Mul(const b2Transform& T, const b2Vec2& v) {
  b2Vec2 result;
  result.x = (T.q.c * v.x - T.q.s * v.y) + T.p.x;
  result.y = (T.q.s * v.x + T.q.c * v.y) + T.p.y;
  return result;
}
```

Scalar Totals
4 muls
3 adds
1 sub
8 total ops



SIMD Totals
1 mul
2 adds
1 sub
6 shuffles
10 total ops

```
result.x = Temp0 – Temp2  // (1x sub, 1x shuffle)
result.y = Temp1 + Temp3  // (1x add, 2x shuffle)
```

Box2D Profiling

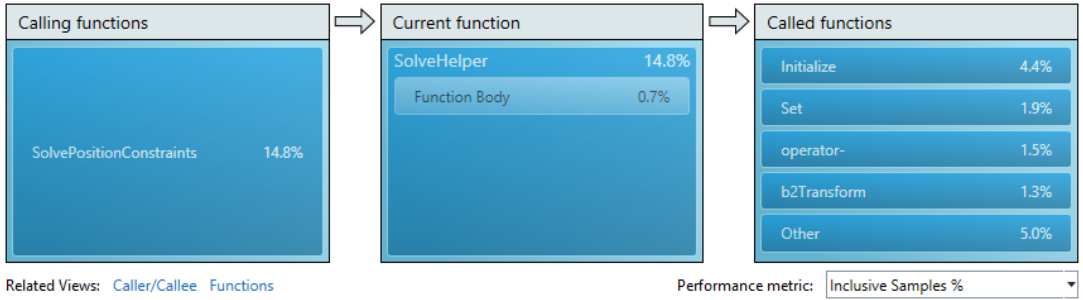
ballpit141013(1).vspx | ballpit141013.vspx | ballpit.h | ballpit_main.cpp | b2Settings.cpp | b2ContactSolver.cpp

Current View: Functions

Function Name	Inclusive Samples	Exclusive Samples	Inclusive Sample...	Exclusive Samples %
b2World::Step	18,867	1	100.00	0.01
__tmainCRTStartup	18,867	0	100.00	0.00
main	18,867	0	100.00	0.00
mainCRTStartup	18,867	0	100.00	0.00
ballpit	18,866	0	99.99	0.00
b2World::Solve	14,897	282	78.96	1.49
b2Island::Solve	7,167	32	37.99	0.17
b2ContactManager::FindNewContacts	5,711	0	30.27	0.00
b2BroadPhase::UpdatePairs<b2ContactManager>	5,710	17	30.26	0.09
b2ContactManager::Collide	3,027	84	16.04	0.45
b2ContactSolver::SolvePositionConstraints	2,805	0	14.87	0.00
b2ContactSolver::SolveHelper				
b2Vec2::b2Vec2				
std::sort<b2Pair *,bool (__cdecl*)(b2Pair const &,b2P				
std::__Sort<b2Pair *,int,bool (__cdecl*)(b2Pair const &(				

b2ContactSolver::SolveHelper

ballpit.exe



If SolvePositionConstraints can be sped up by 3-4x, we can get an overall ~10%

Position Constraint Solver

Sequential Impulse Solver:

- Adjust position of pairwise colliding bodies to eliminate/minimize overlap
- Needs to iterate to get the global solution

```
for each pair of colliding bodies (A, B)
  for each contact point between A and B
    compute position and rotation adjustment
    for A and B, based on:
      1) mass/inertia
      2) center-of-mass/contact point relation
      3) ‘size’ of overlap
```

Position Constraint Solver - simplified

```
bool b2ContactSolver::SolvePositionConstraints() {
  for (int32 i = 0; i < m_count; ++i) {
    b2ContactPositionConstraint* pc = m_positionConstraints + i;
    int32   indexA = pc->indexA;
    int32   indexB = pc->indexB;
    int32   pointCount = pc->pointCount;
    b2Vec2  cA = m_positions[indexA].c;
    b2Vec2  cB = m_positions[indexB].c;
    for (int32 j = 0; j < pointCount; ++j) {
      // A bunch of float32 vector math based on
      // mass, inertia, center-of-mass positions,
      // and contact position
      cA -= mA * P;
      cB += mB * P;
    }
    m_positions[indexA].c = cA;
    m_positions[indexB].c = cB;
  }
}
```

“Holy Grail of Vectorization”

Reduce iteration count by vector width (4)

Enemies of Vectorization

- Control flow in loop
- Data dependencies between iterations

Applied solutions

- Specialize
- Sort data to minimize dependencies within groups of 4

Position Constraint Solver - specialized

```
float32 b2ContactSolver::SimdSolvePositionConstraints() {
  for (int32 i = 0; i < (m_count-3); i+=4) {
    b2ContactPositionConstraint *pc = m_positionConstraints + i;
    int32 indexA[4] = {pc->indexA, (pc+1)->indexA, (pc+2)->indexA, (pc+3)->indexA};
    int32 indexB[4] = {pc->indexB, (pc+1)->indexB, (pc+2)->indexB, (pc+3)->indexB};
    if (IndexOverlap(indexA, indexB)) {
      // doesn't deal with aliasing between the 4 lanes
      COUNTER_INC(indexOverlap);
      float32 minSep = SolveHelper(i, 4);
      minSeparation = b2Min(minSeparation, minSep);
      continue;
    }
    else {
      COUNTER_INC(noIndexOverlap);
    }
    …
  }
}
```

Position Constraint Solver – Sorting Constraints

Without sorting constraints

```
$ ./ballpit simd
indexOverlap:              347472
noIndexOverlap:            2970
Cycles:
SolvePositionConstrains: 6448.29M
SimdSolvePositionConstraints: 6440.18M
Benchmark complete.
  ms/frame: 56.996094
```

With sorting constraints

```
$ ./ballpit simd sortCon
indexOverlap:              631
noIndexOverlap:            348688
Cycles:
SolvePositionConstrains: 2233.05M
SimdSolvePositionConstraints: 2224.84M
Benchmark complete.
  ms/frame: 52.496094
```

Number of overlaps between groups of 4 reduced significantly reduced!

Position Constraint Solver – Sample conversions

Original Code in loop

```
float32 rnA = b2Cross(rA, normal);
```

After manual unrolling by 4

```
float32 rnA[4];
rnA[0] = b2Cross(rA[0], normal[0]);
rnA[1] = b2Cross(rA[1], normal[1]);
rnA[2] = b2Cross(rA[2], normal[2]);
rnA[3] = b2Cross(rA[3], normal[3]);
```

After merging into SIMD

```
__m128 rnA4 = b2Cross4(
                rAx4, rAy4,
                normalx4, normaly4);
```

b2Cross()

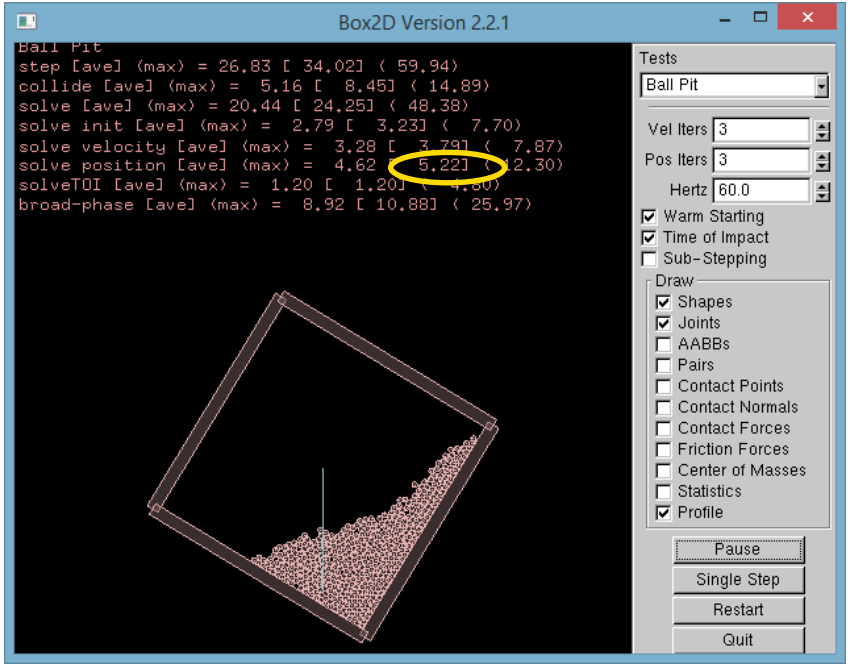
```
float32 b2Cross(const b2Vec2& a, const b2Vec2& b) {
  return a.x * b.y - a.y * b.x;
}
```

b2Cross4()

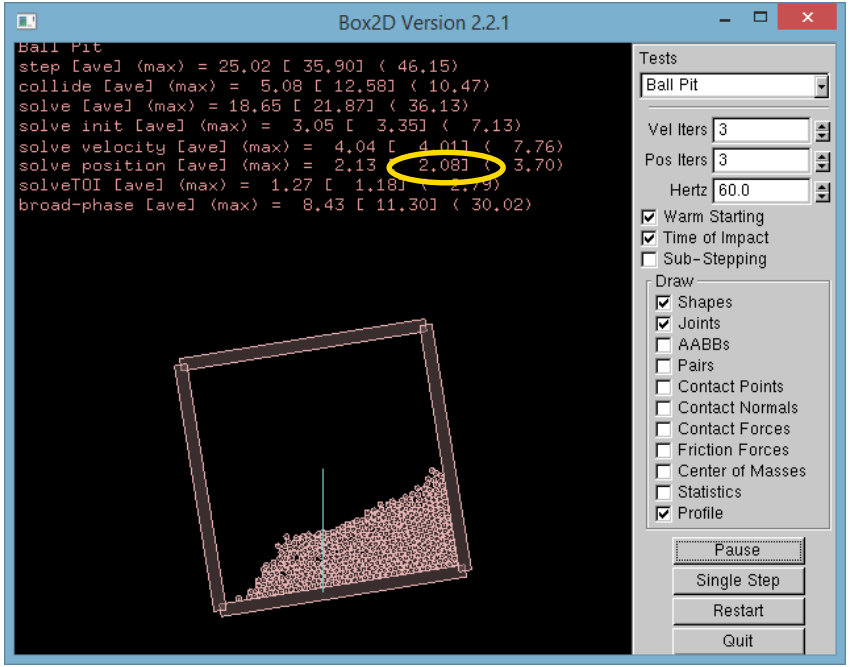
```
__m128 b2Cross4(const __m128 &ax4, const __m128 &ay4,
                const __m128 &bx4, const __m128 &by4) {
   return _mm_sub_ps(
            _mm_mul_ps(ax4, by4), _mm_mul_ps(ay4, bx4));
}
```

Testbed Profiling

Without SIMD



With SIMD



Putting it all together

Use SIMD enabled Emscripten to generate JS
Run in SIMD enabled browser

Summary

Didn't Work:

- Doing SIMDization on leaf functions

Worked:

- Doing SIMDization on Position Constraint Solver, but
  - It requires data restructuring and specialization
  - Overall performance gain is limited

Using Emscripten to generate SIMD.JS from C/C++ is a winner!

- Get gain from already SIMD optimized C/C++ code
- Get portability by using the browser as a platform

Thank You – Questions?

Presentation available here:

http://peterjensen.github.io/html5-box2d