Here is how you create a Compute Pipeline

1. Descriptor Set Layouts
   - VkCreatePipelineLayout()

2. Compute Pipeline
   - VkCreateComputePipeline()
A Reminder about Data Buffers

Creating a Shader Storage Buffer

Vulkan: Allocating Memory for a Buffer, Binding a Buffer to Memory, and Writing to the Buffer

Fill the Data Buffer
And, since we have Data Buffers, we will need Descriptor Sets to Create the Pipeline Layout

Create the Compute Pipeline Layout

Create the Compute Pipeline

The Particle System Compute Shader -- Setup

This is the number of work-items per work-group, set in the compute shader.
The number of work-groups is set in the vkCmdDispatch() function call in the C/C++ program.
#define POINT vec3
#define VELOCITY vec3
#define VECTOR vec3
#define SPHERE vec4

const VECTOR G = VECTOR(0., -9.8, 0.);
const float DT = 0.1;
const SPHERE Sphere = vec4(-100., -800., 0., 600.); // x, y, z, r

uint gid = gl_GlobalInvocationID.x; // the .y and .z are both 1 in this case

POINT p = Positions[gid].xyz;
VELOCITY v = Velocities[gid].xyz;

POINT pp = p + v*DT + .5*DT*DT*G;
VELOCITY vp = v + G*DT;

if (IsInsideSphere(pp, Sphere))
{
    vp = BounceSphere(p, v, S);
    pp = p + vp*DT + .5*DT*DT*G;
}

Positions[gid].xyz = pp;
Velocities[gid].xyz = vp;

The Particle System Compute Shader – The Physics

VELOCITY Bounce(VELOCITY vin, VECTOR n)
{
    VELOCITY vout = reflect(vin, n);
    return vout;
}

VELOCITY BounceSphere(POINT p, VELOCITY v, SPHERE s)
{
    VECTOR n = normalize(p - s.xyz);
    return Bounce(v, n);
}

bool IsInsideSphere(POINT p, SPHERE s)
{
    float r = length(p - s.xyz);
    return (r < s.w);
}

How About Introducing a Bounce?

Dispatching the Compute Shader from the Command Buffer

const int NUM_PARTICLES = 1024*1024;
const int NUM_WORK_ITEMS = 64;
const int NUM_X_WORK_GROUPS = NUM_PARTICLES / NUM_WORK_ITEMS;

vkCmdBindPipeline(CommandBuffer, VK_PIPELINE_BIND_POINT_COMPUTE, ComputePipeline);
vkCmdDispatch(CommandBuffer, NUM_X_WORK_GROUPS, 1, 1);

vkCmdBindPipeline(CommandBuffer, VK_PIPELINE_BIND_POINT_COMPUTE, ComputePipeline);
vkCmdDispatchIndirect(CommandBuffer, Buffer, 0); // Buffer holds the 3 sizes, offset=0

Graphics Trick Alert: Making the bounce happen from the surface of the sphere is time-consuming. Instead, bounce from the previous position in space. If DT is small enough (and it is), nobody will ever know.

This is the number of work-groups, set in the C/C++ program. The number of work-items per work-group is set in a layout in the compute shader.
The Bouncing Particle System Compute Shader – What Does It Look Like?