Project 1: Geometry & Simulation

Due: Fri., Sept 29

Overview. Our first project explores the role of geometry in simulation. Here we will use concepts from geometry, collision detection, and basic physical simulation to create a pinball simulation. The project is split into two parts: the core pinball simulation and a full featured pinball game. The full game (part 2) is optional for undergrads, but required for graduate students.

You may work with a partner on both parts, one project turn-in is needed per pair. But note, that you cannot use the same partner on Projects 2 or 3.

Part 1: Pinball Simulation [up to 100 points *(up to 110 for grad students)*]
You will need to write a visual simulation of multiple balls moving in a pinball style setup. This means the balls need to bounce off of each other and the environment. For part 1, you don't need user interaction.

Required components are indicated with a star (*).
Recommended components are indicated with a plus (+), but feel free to do other options instead if you'd like to focus on different aspects of the project.

Basic Pinball Dynamics* (up to 30 points).
Simulate a ball falling down a screen (accelerating under gravity) that has natural bouncing interactions with multiple objects as it falls down.

Multiple Balls Interacting* (up to 20 points).
Simulate multiple balls interacting with each other all they fall down bouncing through a set of obstacles in the style of pinball machine. The balls need to clearly bounce off each other in a natural fashion. For full credit, the balls cannot penetrate any obstacles (even briefly) or miss collisions, and all motion must look natural.

Circular Obstacles* (up to 10 points).
The pinball simulation must include multiple circular objects that the ball interacts with, in a smooth and natural fashion.

Line-Segment and/or Polygonal Obstacles + (up to 10 points).
Include multiple line-segment and/or polygonal obstacles that the ball interacts with, in a smooth and natural fashion.

Particle System Effects (up to 10 points).
When a ball triggers some event (e.g., rolling over a special tile), trigger a special animation effect with a particle system (e.g., fireworks or explosions). For full credit,

the resulting animation must look well-tuned and appear clearly triggered by the balls' action.

Plunger/Launcher to shoot balls (up to 10 points).
Launch the balls into the scene in the style of a traditional pinball game, making sure the balls have a natural initial velocity and resulting path.

Textured Background+ (up to 5 points).
Use textures or sprite to display an image on the pinball table/background. For full credit, this image should make sense with the placement of any obstacles.

Textured Obstacles (up to 5 points).
Use textures or sprite to display an image on some of the obstacles in the simulation.

Reactive Obstacles (up to 5 points).
Have obstacles that light-up, move, or react in response to the motion of the balls.

Sound Effects (up to 5 points).
Have sound effects that are triggered by the balls' interaction with objects.

Score Display (up to 5 points).
Track some score based on the ball's motion and display this score graphically. (No credit for displaying the score on the command line.)

Multiple Material Types (up to 5 points).
Have objects with drastically different interaction mechanics based on their material properties or other physical simulation properties. For example, this can be areas that are extra bouncy or that add energy through a "kick-back" effect. Each different material must look different visually as well as act differently.

Loading Scenes from Files (up to 10 points).
Create a scene format that allows you to specify the layout of the pinball obstacles; then load the scene from a file. For full credit, you must demonstrate loading at least two different scenes/table layouts.

3D Rendering & Camera (up to 10 points).
Render your simulation in 3D (only the rendering needs to be 3D, the physics can be 2D). For full points the camera should move naturally, the models easy to see, and the scene well-lit with a clear sense of depth. Texturing your models, and using multiple light sources is a good way to achieve this highest level of visual quality.

Progressive Objectives (up to 5 points).
Implement an objective system where the user has different objects as targets that they must chain together to unlock some extra reward or effect. E.g., a series of targets that pop-up after each is hit, areas on the floor that must be hit in a certain order to unlock multi-ball, or maybe knocking over three special targets triggers a

special effect or triggers a particle system animation and gives a score multiplier for the duration of the animation.

Part 2 – Pinball Game [20 point (grad*), 10 points (undergrad)]
The challenge simulation is required if you are a graduate student, and optional extra credit for undergraduates. If you complete the challenge, you have a 72-hour extension on the project.

Description (up to 20 points).
Add (at least) two working flippers, that can be independently triggered by users using the mouse, keyboard, or touch events. The flipper hitting the ball must import momentum/velocity on the ball based on the rotational velocity of the flipper and the point of impact. Integrate this user interaction together with your physics simulation and obstacles from Part 1 to create a complete pinball game. The game must track the player's score (displayed on the console output is fine), support multiple balls, and end when all the balls are lost.

Art Contest
If you generate a pretty image (even by accident), save it to submit to the class art contest. The art will be shared with the class. A pool of honorable mentions will be given 2 points, and the grand winner gets 5 points. All winners will be chosen *completely subjectively*.

Project Report & Video* (10 points).
Your submission must be in the form of webpage with:

- Images of your simulation and/or game
- One or more videos showcasing features of your simulation
- An explicit list of which features you attempted, along with a brief description of the features and timestamp of where it occurs in your video(s).
- All code for your project with a clear indication of what code you wrote
- List of the tools/library you used
- Brief write-up explaining difficulties you encountered
- Submission for the art contest (optional)

These 10 points for the submission itself will be based on the clarity of expression of the report, and the degree which it quickly communicates what you tried, what worked well, and what didn't.

Each feature you expect to get credit for ***must*** be documented in your submission videos in a way which clearly shows the resulting behavior. If you do not show a feature in your submission video(s) you will not receive credit for it.

Grading Criteria
Simulations and/or gameplay must animate well and look convincing to get full credit. Partially implemented features will receive partial credit. Points past those needed for full credit will count as extra credit, though at a discounted rate (see Scoring below). If you do other things that you think are cool and worth credit let us know beforehand and be sure to document it in the report.

Project Scoring
Undergrads may submit up to 120 points of work subject to the following limits:
100 for part 1
10 for part 2/challenge
10 for the report *[required]*
… if you submit more than the limit, we will grade a random subset.

Graduate students may submit up to 140 points of work subject to these limits:
110 for part 1
20 for part 2/challenge *[required]*
10 for the report *[required]*
… if you submit more than the limit, we will grade a random subset.

Partial credit will be given. Scores computed as follows (points above 100 possible):
-*Undergraduate*: Grade is √(totalPoints * 100) [e.g., 100 points will be full credit]
-*Grad students*: Grade is √(totalPoints * 84) [e.g., 120 points will be full credit]

*Extra credit will be given only to projects with an A- or higher on required features.

Use of other code and tools
Anything you are getting credit for must be code you wrote for this course. You must write all of the code for the simulation yourself! This includes both physics simulation code and any collision detection code (you may use an external matrix and vector math library). Learning how to work with external geometry libraries and physics simulation is a very useful skill but it will not count towards this assignment. Likewise, finding fully working simulation code from the internet may be useful for future personal projects, but to receive a grade for this assignment you must turn in your own simulation code you wrote yourself. External libraries may be used for aspects that are not related to reparenting geometry, collision detection, or simulation (e.g., rendering, camera motion, video capture), but just be sure to document that you used these.

Partners & Groups
You are strongly encouraged to work in pairs for the project. Each pair should turn in only one assignment. Both people will be given the same grade. You cannot repeat the same partner on a subsequent project.

If you need help creating a webpage, many online resources exist. UMN's Google Site: https://sites.google.com/a/umn.edu is a great place to start, especially if you have never made a webpage before.