Modern Computer Games

COMP 521, Fall 2025

Assignment 2

Due: Wednesday, October 1, 2025, by 23:59:59
Automatic Extension: Monday, October 6, 2025, by 23:59:59

Note: Late assignments will only be accepted with prior **written** permission of the instructor. Please make sure your code is in a professional style: **well-commented,** properly structured, and appropriate symbol names. Marks will be very generously deducted if not!

Description

This assignment requires you implement a demonstration of some basic game physics in Unity.

As with the previous assignment, aesthetics are not a factor in grading—solid objects should be opaque rather than wire-frame, but you do not need to use external models or textures; creative use of basic assets and asset-combinations can be used to accomplish all objectives.

Overall design: The assignment requires your construct a simulation of a simplified pinball table. The main, static play area is a rectangular region, bounded on all sides, with 4 fixed objects—two "paddles" (triangular shapes pivoting at the base, slightly angled downward toward the centre), each adjacent to a thin rectangle, also angled downward. The intent is to ensure that a pinball rolling down toward the bottom of the table is guided toward the gap between the paddles and can fall into the "gutter" region. The paddles themselves should be positioned such that there is horizontal gap between them (marginally) large enough room for a pinball to fall between them for any rotation of the paddles. Use a fixed camera view giving a clear but perspective view of the table, as shown below.

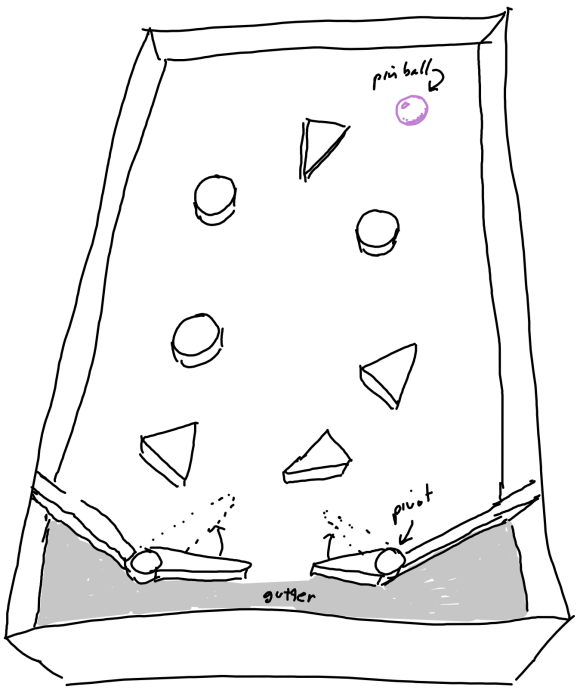

Figure 1: Level sketch.

Specific requirements

1. Build the basic pinball table as a surface with paddles and barriers as described above. Paddles are user controllable—pressing the "a" key flips (pivots) the left paddle up about 45° while held, and the "d" key does the same for the right paddle. Paddle movement should be rapid, but not instantaneous. The "z" key jiggles the table (adding a small amount of random noise to the pinball's movement). **5**

   Sufficient ambient or other light should be present so everything is visible.

2. Populate the pinball table with 3 cylinders and 4 triangular prisms. These objects should be placed randomly, but guaranteed to not overlap and to allow a pinball to move freely between all objects (including the boundary). This randomized placement should be different on every play. Cylinders need to be larger than the pinball diameter, and triangular prisms should not have an edge parallel to the x-axis. **5**

   Add a pinball. This is a sphere, marginally smaller than the gap between the paddles (at rest). The pinball is initialized to a random location by the player pressing the space-bar.

   The pinball table is presumed to have a slight tilt, so the ball is subject to a constant acceleration down, toward the gutter area. A pinball that enters the gutter area disappears.

3. The pinball will encounter objects as it moves around the table, and thus needs appropriate collision detection and response. The different objects surfaces, however, behave slightly differently:

   (a) The pinball should be able to collide with the objects on or bounding the table. Collision with the boundary, lower barriers, or unmoving paddles should result in a bounce with a little energy loss; collision with a cylinder should add energy to the bounce, accelerating it by a noticeable amount; collision with a triangular prism should remove energy from the bounce, roughly in balance with the acceleration provided by cylinders. **5**

   (b) Collision with paddles while they are in motion (pivoting) should give an acceleration to the pinball in proportion to the speed of their movement toward the pinball at the point of collision. **5**

4. Add the ability to have 2 pinballs in play at the same time. Pinballs should be able to collide with each other. **5**

Collisions should appear physically realistic. Bound the speed of the pinball to avoid it moving invisibly fast. You do not need to model spin, and should allow at most two pinballs in play at a time. The simulation should continue until the player terminates the application.

What to hand in

You must submit your **well-commented and readable** Unity source code, along with a **separate document** providing a concise textual description of the state of your solution, how you handled the paddle and pinball-pinball collisions, and describing any missing features or limitations.

For the Unity code itself, hand in a zip file of your project code containing all files needed in order to reconstruct and run your simulation. Please avoid excessive use of assets, especially high-resolution ones, as it easily results in very large archives. Allow ample time to upload your assignment solution, and please follow the general Unity instructions given in MyCourses.

For non-Unity questions, submit either an ASCII text document or a .pdf file *with all fonts embedded*. Do not submit .doc or .docx files. Images (plots or scans) are acceptable in all common graphic file formats.

Assignments must be submitted on the due date **before midnight**. Submit your assignment to *MyCourses*. Note that clock accuracy varies, and late assignments will not be accepted without a medical note: **do not wait until the last minute**.

This assignment is worth 15% of your final grade. **25**

Programming assessment

	Mastery	Proficient	Developing	Beginning
Correctness	The solution works correctly on all inputs and meets all specifications.	The solution meets most of the specifications; minor errors exist.	The solution is incorrect in many instances.	The solution does not run or is mostly incorrect.
Readability	Well organized according to course expectations and easy to follow without additional context.	Mostly organized according to course expectations and easy to follow for someone with context.	Readable only by someone who knows what it is supposed to be doing.	Poorly organized and very difficult to read.
Algorithm Design	The choice of algorithms, data structures, or implementation techniques is very appropriate to the problem.	The choice of algorithms, data structures, or implementation techniques is mostly appropriate to the problem.	The choice of algorithms, data structures, or implementation techniques is mostly inappropriate to the problem.	Fails to present a coherent algorithm or solution.
Documentation	The solution is well documented according to course expectations.	The solution is well documented according to course expectations.	The solution documentation lacks relevancy or disagrees with course expectations.	The solution lacks documentation.
Performance	The solution meets all performance expectations	The solution meets most performance expectations	The solution meets few performance expectations	The solution is not performant.