SD4 Assignment 1: Raycasts vs. Convex Scene

# Requirements

* (18) A 2D scene composed of N (initially 8, say) random static (non-moving) convex objects, drawn nicely
* (15) M (initially 1024, say) invisible random raycasts (from a random position to a random position) per frame
* (18) One visible raycast drawn, with entire ray, first-impact position, and surface normal drawn nicely
* (5) One object (at most) that the mouse position is inside of is highlighted as the “current” object
* (14) Implementation of at least one spatial hashing and/or partitioning scheme (BSP, BVH, bit-buckets, etc.)
* (14) Keyboard / mouse shortcuts for each of the following, all displayed in-app / onscreen:
  + Hold ‘S’ to snap (drag) the visible raycast Start to the mouse; hold ‘E’ to snap (drag) the raycast End
  + Keys (default: ‘.’ and ‘,’) to hit to double and halve (minimum 1) the number of static scene objects
  + Keys (default: ‘M’ and ‘N’) to hit to double and halve (minimum 0) the number of invisible raycasts
  + Keys (default: ‘W’ and ‘R’, or ‘Q’ and ‘E’) to hold to rotate (clockwise and CCW) the current object pointed-at, **rotated around the mouse**
  + Keys (default: ‘L’ and ‘K’) to hold to scale (inflate/deflate) uniformly the current object pointed-at, **scaled around the mouse**
  + Left mouse click-and-drag to drag the current object around;
    - Object should not “snap to” cursor; preserve object’s relative offset from grabbed position
    - Object should not “lose focus” when dragging past/over others; “current” object is locked
    - Only works if there was a “current” object at start of drag (LMB-down); otherwise, does nothing
  + Key (default F8) to re-randomize all shapes & rebuild the entire scene
  + *Keyboard/mouse controls may vary from these, but must be documented clearly in ReadMe.txt*
* (6) Metrics / variables also displayed onscreen:
  + Number of objects, number of raycasts, raycasts total milliseconds, partitioning scheme on/off, FPS, etc.
* (10) A clear and succinct (but not academic-formal) summary of findings (in ReadMe.txt):
  + How many raycasts/ms can you do vs. 100 objects? 1000? 10000?
  + How does this improve (or worsen!) when you enable your hashing/partitioning scheme?
  + How do these speeds compare in each build configuration (Debug, DebugInline, FastBreak, Release)?
  + Any general trends you can observe, i.e. the speed seems to be O(N) or O(N2) with #objects, #rays, etc.
  + Any data specific to your hashing/partitioning scheme you can observe (e.g. AABB Tree depth)
  + Anything else interesting you observe about your results
  + All speed measurements (other than cross-build comparisons) are taken in Release builds

# Submission

* Use your own Guildhall “SD” C++ engine – maintained, buddy-tested, and submitted via Perforce.
  + A committed changelist in P4 with submission comment “SD4-A1: COMPLETE”.
  + Be sure to include an updated ReadMe, a current Release-built .EXE, and all required code & data files.
  + *See notes from Professor Service’s DFS2 class regarding maintenance of your code across P4 / GitHub.*
* Submit a **.zip file** to Canvas named **C29\_SD4\_*A\**\_*p4username*.zip** (*for example:* ***C29\_SD4\_A1\_beiserloh.zip***) which contains:
  + A very short (informal) voice-narrated **video** demonstrating the full functionality of your project;
  + A copy of **this document**, with completed items highlighted cyan, omitted items highlighted red, and partially completed items highlighted yellow (with inserted bullets-text underneath explaining).