* Use the renderer we provided you as the base for this project.
* However, you could use your own pre-created engine instead.
* There are no late fees in this course since the labs depend on each other. But you should try to keep up with the turn-in times or you will fall behind fast.
* Implement update functionality - 10%
  + In the Dev\_app.cpp, there is an update function that you can use.
  + All update and logic happens here(unless you are using your own engine or created a different update function elsewhere).
  + The things you could update are these: updating the velocity of the particles, spawning particles, moving the camera, collecting input, color update changes, ext. Do not use the draw\_view function for those behaviours, as that is for rendering only.
  + Implement deltaTime and use it appropriately.
* Implement the Debug Renderer - 20%
  + Using the debug renderer, you will render a grid (simulating a floor).
  + You will need to call debug\_renderer::add\_line for every line in the grid every frame.
  + Call debug\_renderer::clear\_lines at the end of every frame.(currently already being done)
  + Cycle/change the colors of the grid based on time.
  + The rendering portion is already being handled. It is using the line\_verts from the debug renderer to draw using D3D11\_PRIMITIVE\_TOPOLOGY\_LINELIST.
  + Once you get the debug renderer working, a green checkbox should show on your screen. (the checkbox is being drawn from the dev\_app.cpp.
* Pools
  + Your pools will be used to spawn simple particles.
    - struct particle

{

float3 pos;

float3 prev\_pos;

float4 color;

//could add lifetime if you’d like

};

* + You will need to create a sorted pool and a free pool and use them to spawn your particles.
  + 1 emitter using 1 sorted pool will be used to emit particles.
  + 4 emitters using 1 free pool will be used to emit particles as well.
  + First, implement and use the sorted pool, after it works, then you work on the free pool.
  + For each update:
    - Spawn a few particles (preferably using delta time)(don’t spawn them all)
    - Update living particle positions
    - Free particles based on conditions you decide
      * Ex: Particle falls below grid or ran out of life time
  + The particles will be line segments, use your debug renderer to draw them.
* Implement and Test Sorted Pool - 30%
  + Perform update and rendering for all active elements in the pool.
  + If particle ‘i’ is freed, don’t skip update and rendering for the particle that takes its place.
* Implement and Test Pool with Free List - 30%
  + by using multiple ‘emitters’ that share the pool.
  + Emitters must track the particles they have allocated.
  + Ex:

pool\_t<particle, 1024> shared\_pool;

struct emitter

{

float3 spawn\_pos;

float4 spawn\_color;

// indices into the shared\_pool

sorted\_pool<int16\_t, 256> indices;

};

* Simple Particle behaviour - 10%
  + Have logic that will spawn particles little at a time, to have a constant spray, not fire them all at once. (so the first frame should not have the max particles spawned, it should have maybe 3 spawned. Next frame 3 more, and so on).
  + deltaTime = delta time between updates (in seconds, so it’s usually 0.01)
  + Gravity = Vector3(0, -9.8f, 0)
  + Velocity += Gravity \* deltaTime;(Velocity is a variable in the particle)
  + Pos += Velocity \* deltaTime;
* When you turn your labs in, please zip up the entire project and send it. If you are using your own engine, please be sure to have a working exe in the project as well, as I will likely not be able to build your engine.