

Performance Profiling Report

CPU

We noticed that the constant polling of our title led to some inefficiencies:

WorldSystem::step	580 (4.82%)	0 (0.00%)	salmon.exe
D:\Documents\School\team06\src\world_system.cpp:122			
	128	int screen_width, screen_height;	
	129	glfwGetFramebufferSize(window, &screen_width, &screen_height);	
	130		
8 (0.07%)	131	checkIfKnightIsMoving();	
36 (0.30%)	132	animateKnight(elapsed_ms_since_last_update);	
499 (4.14%)	133	updateWindowTitle();	
	134	levelCompletionCheck();	
	135	resolveMouseControl();	
26 (0.22%)	136	stuckTimer(elapsed_ms_since_last_update, screen_width, screen_height);	
7 (0.06%)	137	invincibilityTimer(elapsed_ms_since_last_update);	
2 (0.02%)	138	handlePlayerOneAttack(elapsed_ms_since_last_update);	
	139	handlePlayerTwoProjectile(elapsed_ms_since_last_update);	
	140	deathHandling();	

By only updating the title when necessary, we can improve performance:

WorldSystem::step	39 (0.55%)	1 (0.01%)	salmon.exe
PhysicsSystem::wallCollides	37 (0.53%)	4 (0.06%)	salmon.exe
D:\Documents\School\team06\src\world_system.cpp:122			
	128	int screen_width, screen_height;	
	129	glfwGetFramebufferSize(window, &screen_width, &screen_height);	
	130		
5 (0.07%)	131	checkIfKnightIsMoving();	
21 (0.30%)	132	animateKnight(elapsed_ms_since_last_update);	
1 (0.01%)	133	levelCompletionCheck();	
	134	resolveMouseControl();	
5 (0.07%)	135	stuckTimer(elapsed_ms_since_last_update, screen_width, screen_height);	
3 (0.04%)	136	invincibilityTimer(elapsed_ms_since_last_update);	
2 (0.03%)	137	handlePlayerOneAttack(elapsed_ms_since_last_update);	
	138	handlePlayerTwoProjectile(elapsed_ms_since_last_update);	
	139	deathHandling();	

Memory

There appeared to be a significant memory leak in our game:

Native Memory (salmon.exe)

Object Type	Count ▾	Size (Bytes)
salmon.exe!Motion	17,390	486,920
salmon.exe!std::_Container_proxy	252	4,032
salmon.exe!std::_List_node<std::pair<unsigned int const,unsigned int>,void *>	117	2,808
salmon.exe!std::pair<int,int>[]	65	1,040
char[]	34	2,112
salmon.exe!std::_List_unchecked_iterator<std::_List_val<std::_List_simple_types<std::pair...	33	33,933
salmon.exe!glm::vec<2,float,0>[]	22	560
salmon.exe!Entity[]	21	4,032
salmon.exe!ColoredVertex[]	11	5,976

The excessive memory usage in the heap appears to be caused primarily by a single line:

```

115 void AISystem::stepEnemyChase(float elapsed_ms) {
116     // update enemy chase so it chases the player
117     for (Entity entity : registry.enemyChase.entities) {
118         auto& enemyCom = registry.enemies.get(entity);
119         Motion& motion = registry.motions.get(entity);
120         Motion& motion_wz = *new Motion();

```