Technical Guide

Milestones 4

Team Name: _____
Game Name: _____
Milestone: _____

Total points: 170

1) Describe the overall architecture of this project. Include at least one diagram. (Example systems: Core, Factory, Physics, Graphics, Audio, etc.) Be sure to include file names in your diagram to help illustrate where the features live. (20)

2) What are the major technologies that are being used? (Example: DirectX 9.1, FMOD Designer, etc.) (5)

3) What 3rd-party libraries and assets are being used? Abide by the copyright notice and list them, along with their sources. (5)

4) List the style conventions that are being used in this project for code formatting, documentation, file naming, etc. (10)

5) Describe/show the support you have for debugging your game and how they are used: (10)
- Do you have an in-game debug console or output-only console?
- Do you have debug drawing systems?
- Do you have a clean assertion system?
- Do you have an in-game performance visualizer?
- Do you have a way to watch variables in-game?

6) What other debug features has your team implemented? (+5)

7) How is debugging enabled/disabled? (10)

8) Explain your graphics API. Are you using fixed-function or shaders? (10)

9) Explain the process for loading graphical assets, such as sprites, models, textures, animations, etc. (10)

10) Explain any process that your game might use to load and parse other game data, such as level files, enemy definitions, etc. (10)

11) Describe your physics API and processing from within your game engine.
What kind of integration does your Physics API rely on? (Euler, Improved Euler, Verlet, Runge-Kutta, etc.) (10)

12) What kind of space partitioning is used? (5)

13) Explain collision detection in your engine. (10)

14) Explain your project's AI systems and how they inform AI behavior. (10)

15) What types of algorithms were used to implement this behavior? Please describe any of the following features your game incorporates, in regards to AI: (10)
- Pattern movement
- Pathfinding
- State machines
- Flocking
- Influence maps
- Other algorithms

16) Explain any networking that your engine supports and uses for this game. What is the method for hosting and connecting to games? What kind of network protocols does it use? What features are included in it? (Like encryption, data compression, host migration, rollback techniques, etc.) (+10)

17) Does your game support multiplayer in any way? If so, how? (+10)

18) Explain the content development pipeline for your game. If someone wanted to create another level or more enemies, what would be the process for that? (10)

19) Describe/show all tools that are used in your project's development. (10)

18) Describe/show the structure of how your editors work, along with: (+10)
- How do you create/delete/copy objects?
- How do you move/rotate objects?
- How do you save/load levels?
- How do you update archetypes?

19) Describe/show the audio pipeline. (10)

20) Describe/show the art pipeline. (10)

21) Does your engine leverage any kind of scripting langauge? If so, then please describe/show that! (+10)

22) For the following features, please mark if they were implemented within your project. (0)

Feature	Implemented?
Voice-over audio	
In-game SFX	
Background music	
Spatial audio	
UI/menu SFX	
Advanced/dynamic audio filters	
Dynamic lighting/shading	
Vector graphics	
Sprite-based animation system	
Sprite scaling	
Sprite rotation	
Particle effects	
Image masks	
Partial-transparency image blending or alpha blending methods	
Parallax backgrounds	
Multiple layers of background graphics	
Kinematic/skeletal art	
Scripted motion using vector paths	
Animation tweens	
Multiple levels/environments	
Advanced physics simulations	
Local multiplayer	
Networked multiplayer	
Downloadable content or online content fetching	

Any other network features	
Integration into web technologies (Such as web portals)	
Component-based architecture	
Game objects use C++ interfaces	
Game objects are data-driven from factories	
Menu systems	
In-game HUD	
File parsing for gameplay content, such as levels, scenarios, enemies, etc.	
In-game level editor	
Scripting language integration	
Metaclasses, reflection, data-binding	
Live object property inspection	
Art pipeline tools	
Audio pipeline tools	
Engine/game build (compilation) tools	
External gameplay editor	
In-game gameplay editor	
Testing tools and advanced debugging features	
Ability to jump into the game in a given scenario for testing	
Fast-forward and rewinding gameplay	
Extensive/complex debug drawing	
In-game performance visualization	