# Design Goals

Phase 1

* Build a “code challenge” game in C#, where developers can add new characters to the competition by uploading a DLL.
* To add a character, developers implement an interface and consume a simple API for interacting with the game world.
* A governor observes characters and ensures that they are constructed within certain guidelines.
* To start, the only game mode is deathmatch, where it’s every bot for himself.
* Characters move around on an open 2D roguelike board (libtcod?).
* All terminology (including API names, and the names for things like the governor, and error messages from the API) should be highly thematic and optimized for fun!

Phase 2

* Multiple heroes in one DLL are on the same team, greatly opening up design versatility.
* Governor will be extended to provide team management.
* Communication API for coordinating between teammates.
* More game types. Quest mode? Capture the flag?

# Character Creation

We don’t want anyone tampering with stats after the governor approves them, so there should be an API for setting up the sensitive parts. So your character might contain several lines like this:

API.CharacterCreation.SetStat(Character, Stats.Strength, 5);

API.CharacterCreation.AddEquipment(Character, Equipment.Axe);

…And then after your character creation method executes, the manager checks it and locks it to prevent any more character creation API calls.

# Character Actions

Each turn, the character’s DetermineAction() method gets called and the character can return one action. Characters will interact with the world using an API.

Minimal set of actions:

* **API.Actions.Move (direction):** moves to another square, if possible.
* **API.Actions.Attack (target):**
* **API.Actions.Use (item):** maybe drink a potion or whatever.
* **API.Actions.Say (words):** mostly for the benefit of humans who are watching.
* **API.Actions.Pass():** do nothing.
* **API.Actions.Suicide():** quit.

The action is an object that gets passed back into the engine, where it is evaluated for validity and executed.

Evaluation  
I for evaluating surroundings, other characters, and items:

* **API.Evalulation.Look():** returns an array of nearby spaces with coordinate’s relative to the character, the space’s contents (people and things), whether spot is visible, and whether the spot is traversable.
* **API.Evalulation.Examine():** returns visible details of a person or object. We should keep this as simple as possible.

...Maybe an event-driven API also. For instance you probably want to be able to react if someone attacks you, and you will want to know whether they were able to hurt you.