ICPC Game AI Challenge

Powered By

DEVIATION GAMES

Contestant Handbook

icpc | university commons

icpc.foundation

Table of Contents

Changelog

Revision	Notes
r009	Initial version.
r010	macOS documentation (section 2 and 3). Expanded the FAQ. Windows UE4 prerequisites installers (section 2).
r011	macOS notes and Ubuntu setup notes. For FAQ.
r012	Added 6.1 Problem Spec Addendum to note bugs and differences from the spec. Clarified that you can have frames that are 16ms on average.
r013	Added new realtime visualizer (5.6). Organized FAQ by revision.

1. Introduction

In the ICPC Planetoids Challenge, powered by Deviation Games, your submission will serve as a controller for an agent in a game called *Planetoids*. The problem style operates like a typical ICPC interactive-style problem. Your submission can get the current game state by reading from STDIN and send the controller commands to the agent by writing to STDOUT. The input/output format is basic text (JSON simulation frames in, commands as plaintext out). The agent's actions in the game will result in a score based on the rules of the game, and your objective is to create a submission that controls an agent to achieve the highest score.

For the complete problem description, please see https://planetoids21.kattis.com/problems. This document serves as a User Guide for how to run *Planetoids* locally on your machine so you can quickly iterate on an algorithm before submitting to Kattis.

We kindly ask, to the best of your ability, that you use the ***Local Development Package*** (link to download below) in order to develop and test your submission *before* submitting to Kattis. **Please limit yourself to one submission per hour on Kattis.**

This document may change as issues arise and items need clarification. You can find the latest version on the Discord server (link in section 7).

2. System Requirements

In order to use the pre-packaged *Planetoids* game for testing you will need the following depending on your OS:

- Windows 10, Direct X11, 4GB RAM (8GB recommended), 64-bit AMD/Intel
    - **(Required)** .NET 3.1 Runtime (download)
    - (Recommended) Install Powershell 7 (instructions)
    - (Optional) C++: Visual Studio w/ C++ modules
    - (Optional) Python3: You can install via the Microsoft Store
    - *(Optional) C#: Visual Studio w/ C# modules

- Ubuntu 20 (or similar), Vulcan, 4GB RAM (8GB recommended), 64-bit AMD/Intel
    - **(Required)** Mono: `sudo apt install mono-runtime mono-mcs`
        - NOTE: mono version must be > = 6.8 (run `mono --version` to check)
    - (Recommended) Ensure you have the latest vulkan drivers.
    - (Optional) C/C++: `sudo apt install build-essential`
    - *(Optional) Java: `sudo apt-get install default-jdk`
    - *(Optional) Kotlin: (see: Java) `sudo snap install --classic kotlin`
    - *(Optional) C#: `sudo apt-get install mono-mcs`
    - *(Optional) Objective-C: `sudo apt-get install gobjc gnustep gnustep-devel`
    - *(Optional) COBOL: `sudo apt install gnucobol`
    - *(Optional) Rust: `sudo apt install rustc`

- MacOS 11.4+ (Big Sur), 4GB RAM (8GB recommended), 64 Intel
    - **(Required)** XCode Tools: xcode-select –install
    - **(Required)** Mono Runtime: brew install mono

NOTE: Virtualizing Ubuntu + rendering in Unreal Engine is not expected to work. You can still run the game headless, but you are better off running the game natively on your hardware.

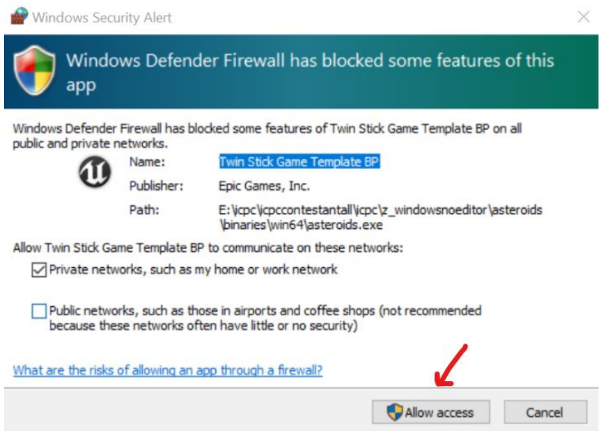
*(Optional) means "You can try but we don't provide Local Development Package support".

3. Setup

You will need to do the following things:

- 1. **Local Development Package**: (IcpcContestantAll.zip) – Download and unzip.
    - This contains the local playable version of the game and template submissions that you can start building from.
- 2.a (**Windows Users**) Do all of the following (to make sure things work):
    1. Run **ICPC/z_win_prereqs/UE4PrereqSetup_x64.exe**
    2. Pick one of:
        a. C++
            i. Open **ICPC/cpp/planetoids.sln** and build via Visual Studio
            ii. Modify **ICPC/SubmissionWrapper.bat** to use the C++ submission.
        b. Python
            i. Modify **ICPC/SubmissionWrapper.bat** to use the Python submission.
    3. Run **ICPC/LaunchGame.bat** (double-click to launch) - the game should open. **KEEP THE GAME RUNNING**
    4. Run **ICPC/LaunchSubmission.bat** (double-click to launch) - the sample AI should start playing the game. **THE GAME MUST ALREADY BE RUNNING**
        a. **NOTE:** If **LaunchSubmission** doesn't work, refer to 5.3 "Process Pipes".

! (Windows) You **must** allow network access for the Socket communication layer to work. If you accidentally decline, you can search "*Allow an app through Windows Firewall*" and delete the settings and re-open the game to get the prompt again.


Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of Twin Stick Game Template BP on all public and private networks.

Name: Twin Stick Game Template BP
Publisher: Epic Games, Inc.
Path: E:\icpc\icpccontestantall\icpc\z_windowsnoeditor\asteroids\binaries\win64\asteroids.exe

Allow Twin Stick Game Template BP to communicate on these networks:

[✓] Private networks, such as my home or work network

[ ] Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

- 2.b (**Linux Users**) Do all of the following (to make sure things work):
    1. Pick one of:
        a. C++
            i. In a terminal, navigate to **ICPC/cpp/** and run **make** to build.
            ii. Modify **ICPC/SubmissionWrapper.sh** to use the C++ submission.
        b. Python
            i. Modify **ICPC/SubmissionWrapper.sh** to use the Python submission.
    2. Run **ICPC/LaunchGame.sh -** the game should open. **KEEP THE GAME RUNNING**
        - **NOTE:** If you are running in a virtual machine, you can also use **LaunchGameHeadless.sh**, you just can't play as a human player or see the results of the run.
    3. Run **ICPC/LaunchSubmission.sh** - the sample AI should start playing a game. **THE GAME MUST ALREADY BE RUNNING**
        a. **NOTE:** If **LaunchSubmission** doesn't work (or is slow), refer to 5.3 "Process Pipes".

! (Linux) Ensure that your executables can run by issuing the following commands:

```
chmod +x ./LaunchGame.sh
chmod +x ./LaunchSubmission.sh
chmod +x ./SubmissionWrapper.sh
chmod +x ./z_glue/Socket/x64/Debug/SocketApp
chmod +x ./z_LinuxNoEditor/Asteroids.sh
```

- 2.c (Mac Users) Follow (2.b) but use the scripts with "_macOS" in their name instead. Also, allow permissions that pop up when you launch the game and the various scripts.

❗ (Mac) Ensure that your executables can run by issuing the following commands:

```
chmod +x ./LaunchGame_macOS.sh
chmod +x ./LaunchSubmission_macOS.sh
chmod +x ./SubmissionWrapper_macOS.sh
chmod +x ./z_glue/Socket/x64/Debug/SocketApp_macOS
chmod +x ./z_MacNoEditor/Asteroids.app/Contents/MacOS/Asteroids
```

I. You will additionally need to right-click "Open" the following apps:
   i. `z_MacNoEditor/Asteroids.app`
   ii. `Z_glue/Socket/x64/Debug/SocketApp_macOS`

II. For the **cpp/** example, copy the contents of **cpp/Makefile_macOS** into **cpp/Makefile.**

❗ (Mac) macOS support is "best effort" and has not been tested nearly as much.

4. Programming Language Support

4.1. Overview

You can theoretically use any language supported by Kattis with the *Local Development Package*. However, the package has only been tested with C++ and Python submissions on Windows and Ubuntu.

In order to use a different language (Rust, COBOL, etc.) you need to install whatever your OS requires and then modify **ICPC/SubmissionWrapper[.sh|.bat]** to launch your submission in that language's preferred way.

NOTE: For information on how to get started with various programming languages in the Kattis system see: https://planetoids21.kattis.com/help

! Your submission **must** compile / be interpreted according to the Kattis system's language requirements. As a shorthand, your submission should compile in an Ubuntu 20 environment with the following packages installed. If you are curious what compiler flags are used for your submission, then look here. **Having a submission that works locally does not guarantee Kattis will accept your solution.**

4.2. Support Matrix

The following is a table that shows which languages Kattis supports and what testing we have done with each. Again, the other languages may work but there may be some unknowns.

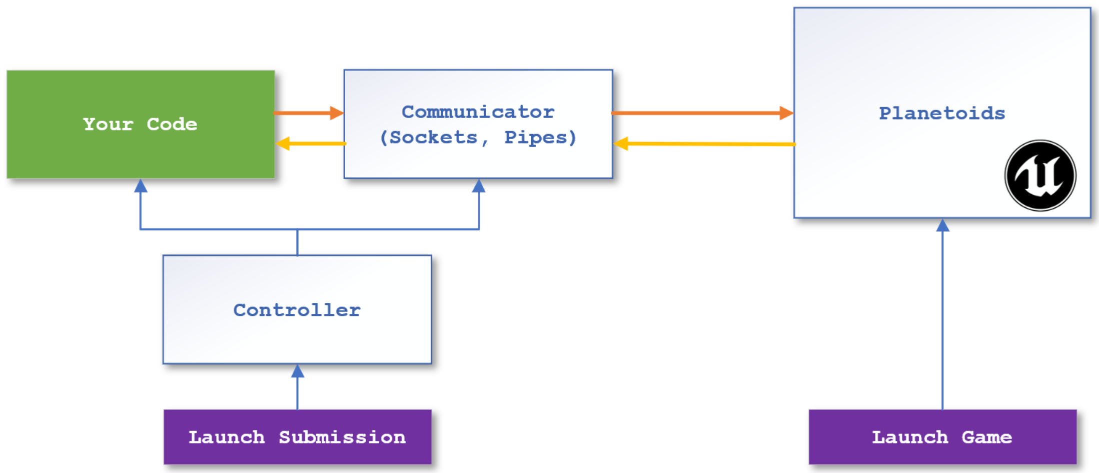
- "Official" means we have tested it locally and in Kattis extensively and have provided an example submission.
- "Unofficial" means we have done a lot less testing, but you at least get an example submission as a starting point.

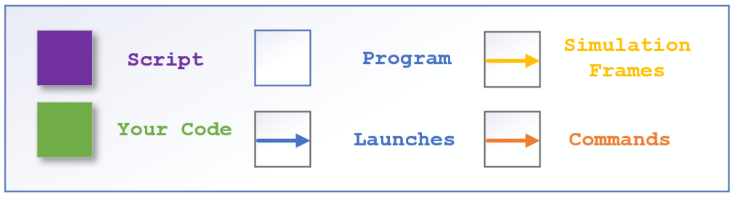
Language	**Support Status**	**Notes**
C++	Official	JSON Parser included.
Python 3	Official	JSON Parser built in.
Python 2	Unofficial*	JSON Parser built in, modify Python 3 template.
Java	Unofficial	No JSON Parser or local package support.
Kotlin	Unofficial	No JSON Parser or local package support.
C	Unofficial*	No JSON Parser or local package support.
Objective-C	Unofficial*	No JSON Parser or local package support.
C#	Unofficial*	JSON Parser may not be available. No local package support.
F#	Unknown	(Look at the C# example and see if you can convert it)
Rust	Unofficial*	No JSON Parser or local package support.
Go	Unknown	
PHP	Unknown	
Prolog	Unknown	
COBOL	Unofficial*	No JSON Parser or local package support.
Haskell	Unknown	
Pascal	Unknown	
Ruby	Unknown	
Node.js	Unknown	
SpiderMonkey	Unknown	
Common Lisp	Unknown	

NOTE: Unofficial* means it should work, but we did not test submissions in those languages on Kattis.

5. Workflow

5.1. How Does the Package Work?





Planetoids can accept input in a variety of ways:

- Input from a Human player (keyboard)
- via Sockets (connected through localhost)
- via Process Pipes (direct subprocess execution)

"**LaunchSubmission**" is a script that will connect your submission to a sockets communicator which in turn connects to the running game instance. When you send a command, it is packaged by the communicator, sent through a socket to the game, and then a simulation frame is returned as a reply. This happens until the game is over.

SEE: "How to run your AI" for more details on how to run via *sockets* or *process pipes*.

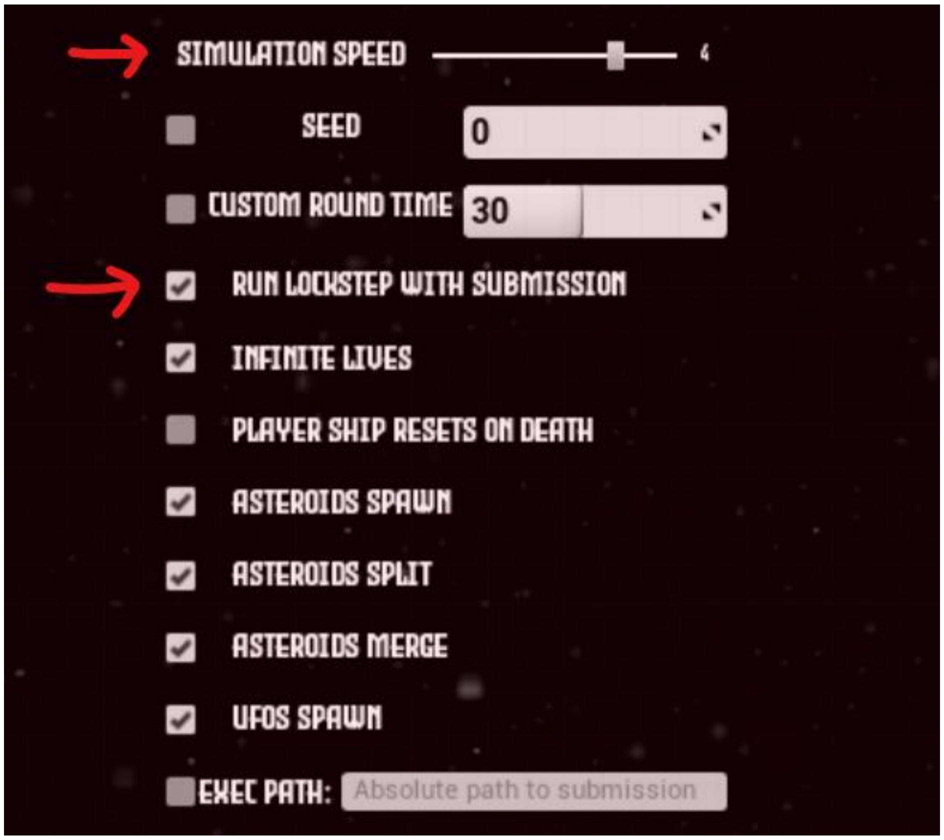
5.2. Considerations When Making Changes to the AI

1. Try to keep all your changes inside of the single submission file, but Kattis ultimately dictates limitations on submission size and number of files (we do not check locally that your submission meets Kattis's requirements).
2. You cannot add other libraries or change compiler / execution flags.
3. You cannot print debug messages to STDOUT. You must use a log file.
4. You can do multithreaded programming, but only plan on one extra thread of execution being effectively available beyond the main thread.
5. The game in the submission environment will run at 4x simulation speed, in lockstep (more on this later).
6. You should not exceed ~~4ms~~ 16ms when processing a single simulation frame (see FAQ for more info). There is an overall and execution time limit for your AI to run, to try to process frames as fast as possible.

NOTE: For C++, the provided *json.hpp* is already present in the Kattis server environment. You do not need to include this in your submission.

5.3. How to run your AI

- Run **ICPC/LaunchGame[.bat|.sh]** - the game should open to the title screen.
  - Go to the Options menu:
    - Toggle on “Run Lockstep with Submission”.
    - As needed, you can change the simulation speed (**4x is the speed you are graded at**, but it is useful to test at 1x speed).



SIMULATION SPEED 4

SEED 0

CUSTOM ROUND TIME 30

✓ RUN LOCKSTEP WITH SUBMISSION

✓ INFINITE LIVES

PLAYER SHIP RESETS ON DEATH

✓ ASTEROIDS SPAWN

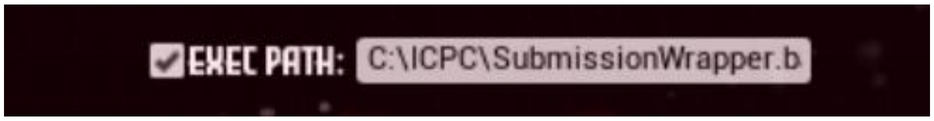
✓ ASTEROIDS SPLIT

✓ ASTEROIDS MERGE

✓ UFOS SPAWN

EXEC PATH: Absolute path to submission

- Run your submission (choose one):
  - **(Socket Communication)** Run **ICPC/LaunchSubmission[.bat|.sh]** - your AI should start playing the game.
  - **(Process Pipes)** In the Options menu set the absolute path to your submission, and then hit “Start Game” in the main menu.

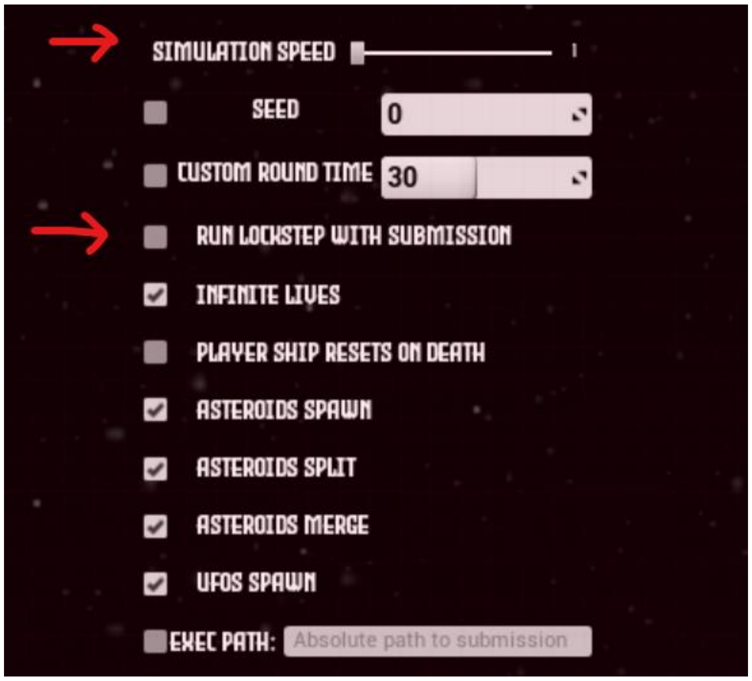


✓ EXEC PATH: C:\ICPC\SubmissionWrapper.b

NOTE: The game supports being connected to and run repeatedly. **LaunchSubmissionRepeatedly.ps1** (Windows-only) does just that so you can run your AI multiple times in a row. You can reset your scores in the Options menu and see the average / high score on the title screen.

5.4. You can also play as a Human

- Run **ICPC/LaunchGame[.bat|.sh]** - the game should open.
- Go to the Options menu and ensure:
    - "Run Lockstep with Submission" turned off.
    - Simulation speed is set to 1x.
    - Disabled the "Exec Path:" option (if set).



SIMULATION SPEED

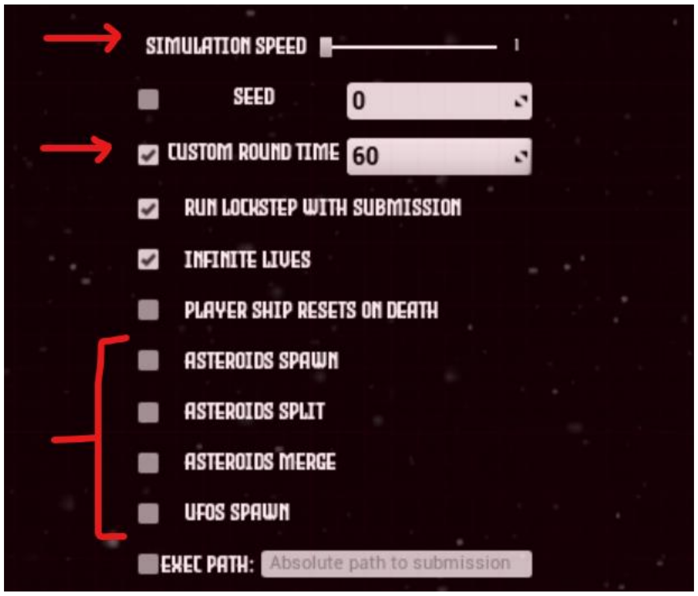
- [ ] SEED: 0
- [ ] CUSTOM ROUND TIME: 30
- [ ] RUN LOCKSTEP WITH SUBMISSION
- [✓] INFINITE LIVES
- [ ] PLAYER SHIP RESETS ON DEATH
- [✓] ASTEROIDS SPAWN
- [✓] ASTEROIDS SPLIT
- [✓] ASTEROIDS MERGE
- [✓] UFOS SPAWN
- [ ] EXEC PATH: Absolute path to submission

- Hit **Play** at the main menu (Controls detailed on main menu)

5.5. Making the Game Easier for Testing / Development

You can make the game easier by:

- Reducing simulation speed
- Increasing the round time
- Disabling asteroids and UFOs



SIMULATION SPEED 1

☐ SEED 0

✓ CUSTOM ROUND TIME 60

✓ RUN LOCKSTEP WITH SUBMISSION

✓ INFINITE LIVES

☐ PLAYER SHIP RESETS ON DEATH

☐ ASTEROIDS SPAWN

☐ ASTEROIDS SPLIT

☐ ASTEROIDS MERGE

☐ UFOS SPAWN

☐ EXEC PATH: Absolute path to submission

GameSettings.ini can be modified to also change how game the runs. You can do things like set defaults, increase the round time, etc.

Parameter	Default	Meaning
`DefaultTimescale`	`1.0`	The simulation speed at startup.
`bRunLockstep`	`False`	Default lockstep when starting the game.
`ScorePath`	`score.txt`	Where to save the score to if want it logged to disk. Specify an absolute path or else it will show up under z_*NoEditor/Asteroids/Saved/
`bLaunchContestantApp`	`False`	Whether or not your submission launches directly and uses process pipes to communicate.
`ContestantExecPath`		Absolute path to the submission to launch when bLaunchContestantApp=True. You can point to "SubmissionWrapper".
`TerminateOnGameEnd`	`False`	Whether to shut down after round finishes.
`Seed`	`0`	The seed used for game randomization.
`RoundLength`	`30.0`	Length of the game in seconds.

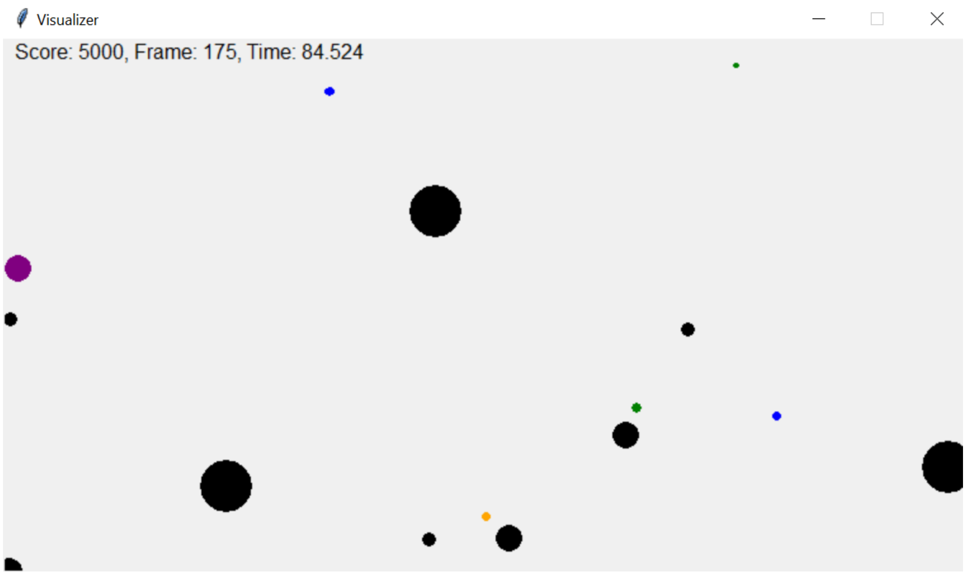
5.6. Simulation Frame Visualizer

This visualizer is "best effort". You are expected to extend this yourself as needed.

z_viz/Visualizer.py is a tkinter python application that polls a file and renders the positions + sizes of all game entities to the screen.

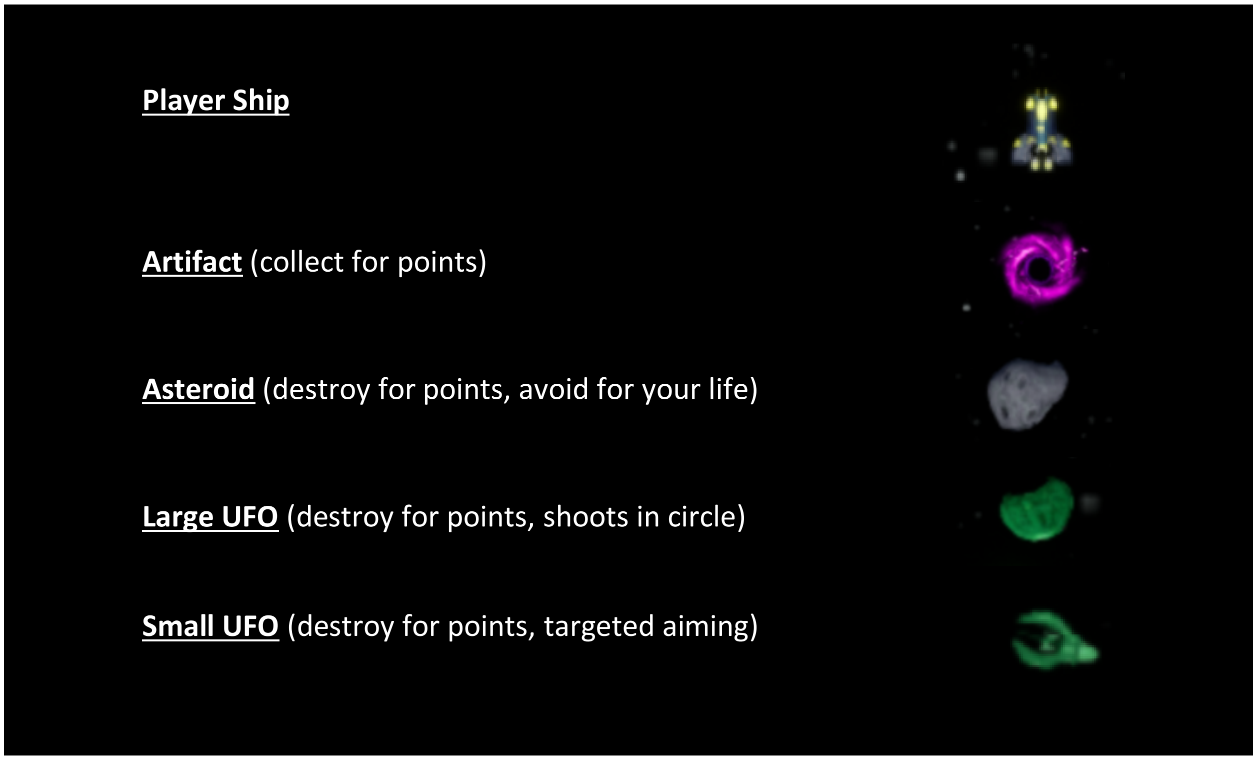
1. Ensure you have python3 (see section 2) and tkinter (**pip3 install tk**) installed.
   a. (Ubuntu) `sudo apt-get install python3-tk`
2. In your submission, create a log that simply writes the JSON to disk. Ensure you flush regularly.
3. Run *Visualizer.py* with the path to where your log should be when it is created.
4. Start the game.
5. Run your submission.

You should see *Visualize.py* start to render the simulation. Look at the code for what colors mean what. **z_viz/planetoids_viz.py** is an example submission that writes the correctly formatted file that *Visualize.py* consumes.



Score: 5000, Frame: 175, Time: 84.524

6. Gameplay Overview


Player Ship
Artifact (collect for points)
Asteroid (destroy for points, avoid for your life)
Large UFO (destroy for points, shoots in circle)
Small UFO (destroy for points, targeted aiming)

A description of the rules is located on Kattis.

6.1. Problem Spec Addendum

What	Radius
Player Ship	32
UFO (Large)	35
UFO (Small)	17
Asteroid (Large)	200
Asteroid (Medium)	100
Asteroid (Small)	50
Bullets	32
Artifact	100

6.2. Bugs / Discrepancies with the Problem Spec

- **bulSrc** is always ‘0’, regardless of the bullet’s source.
- The player’s **Fire Bullet cooldown** is actually 0.5 seconds, not 3.0 seconds.

7. FAQ

Revision 9

How do I quit the game?

One of the following should work:

- Hit the “Escape” key while playing.
- Hit “Quit” on the main menu.
- Hit tilde ~ and then type “quit” at the command prompt.
- Use your OS’s Task Manager to force-quit.

Is there a beginner’s guide to coding for this challenge, and what should I submit on Kattis?

We provide templates for several languages in the *Local Development Package* that lays out the basic loop of the game (read simulation frame, process, emit command, repeat). Look at section 4.1. There is a link to per-language support in the Kattis system. Check the Support Matrix in 4.2 to see any notes about the language you want to write in.

When it comes time to submit, take your file (for example, cpp/planetoids.cpp) and copy it to the Kattis submission area. You do not need to include any other files. Each example is designed so that it should be able to be submitted and successfully graded.

Where do I go if I have questions?

Please visit our Discord server to ask questions and receive the latest version of this document.

Revision 10

What does it mean to run in “lockstep”?

At a high level you are running the game in "lockstep" (it waits for your next command before stepping the game simulation one step). Since the Game has a fixed time it runs (30 in-game seconds), you have an exact number of frames that your submission will be given while it is graded, no matter how long you took to give it a command.

The game itself has its own clock irrespective of the real clock of the hardware it is running on. For example, if you "pause" the game, the game's world time stops ticking, even though real time is passing. So, when you run lockstep, imagine it like the game receives input, steps the simulation one in-game tick, sends out a simulation frame, then pauses and waits for your next command.

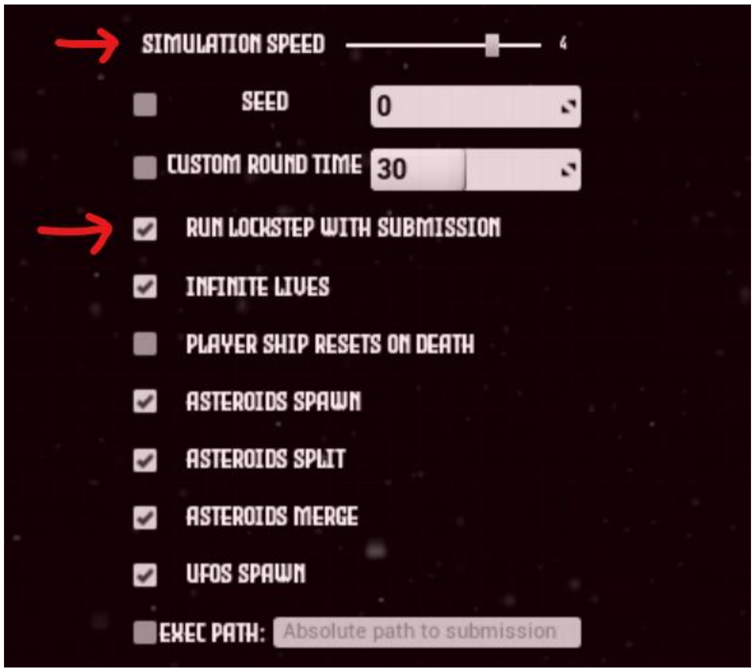
When you say "You should not exceed 4ms when processing a single simulation frame." What does this mean? Is this even correct?

In short: no, we were wrong. You have up to 16ms.

Long answer: This has to do with the rate of simulation (how fast the game ticks and how fast it can process inputs). If you run the game at 1x simulation speed (normal, human-playable), you have 16ms per frame (real time). If you run the game at 4x simulation speed, your frame duration is shorter (4ms). However, we (very late in development) started running the game at a “fixed frame rate” of “60 FPS” to fix issues with determinism when run on Kattis. This causes there to be less frames to be simulation over the course of the game (450 instead of 1800). This means that you do in fact have up to 16ms still per frame on average without your submission getting rejected.

In summary: the real hardware is running your submission at 4x meaning the game should be over in 30/4 = 7.5 seconds. During that time, it should have processed 7.5 * 60 * 4 = 1800 frames. You can take however long you want for each frame but on average you have to give 1800 outputs in those 7.5 seconds, meaning for each frame you should on average not take more time than 7.5s/1800 = 4.1ms. *However*, since we use a fixed-frame-rate we only simulate 1800 / 4 = 450 frames over the course of a game, meaning 7.5s/450 = 16ms.

What simulation settings does the Grading System (Kattis) use?


SIMULATION SPEED 4
SEED 0
CUSTOM ROUND TIME 30
RUN LOCKSTEP WITH SUBMISSION ✓
INFINITE LIVES ✓
PLAYER SHIP RESETS ON DEATH
ASTEROIDS SPAWN ✓
ASTEROIDS SPLIT ✓
ASTEROIDS MERGE ✓
UFOS SPAWN ✓
EXEC PATH: Absolute path to submission

(Ubuntu) What happens if my screen is black?

We do not support running the game in an Ubuntu VM. For native Ubuntu installations, it’s possible Unreal does not have support for your particular GPU x Vulcan installation, in which case you will have to stick to the “RunGameHeadless” scripts. Unfortunately, Unreal's Vulcan driver does not work uniformly across all hardware and OS versions. You should make sure your drivers are up to date. If you have a Windows or Mac machine available, that may be your best bet for getting the GUI version of the game working. However, see section 5.6.

How do I change the resolution of the game?

Open up **LaunchGame[.bat|sh]** and edit the ResX and ResY command-line arguments. If you want to enter fullscreen, simply remove the ResX and ResY parameters.

How many “rounds” will occur during my 30 second game? Will running faster let me have more time to score points?

The concept of a "round" changed during development. For this competition the game has only one "round" (which takes up the whole 30 seconds). For the “running faster” part, the game runs in “lockstep” (see “What does it mean to run in ‘lockstep’” above).

How many lives do I get when being graded?

You have infinite lives. See “What simulation settings does the Grading System (Kattis) use?” above.

Revision 11

Does the Local Develop Package enforce any timeouts?

No, the local package does not enforce timeouts. If you are running in lockstep you can take as much as you want between frames.

How can I interactively play the game through the Terminal?

Run **LaunchGame** normally. Then run

```
z_glue/Socket/x64/Debug/SocketApp 127.0.0.1
```

directly in your terminal. You should now be able to submit commands to control the ship.

(Ubuntu) How to fix LaunchSubmission.sh “get_ArgumentList” not found error?

Ensure you have a version of mono installed that is >= 6.8. Run **`mono --version`** to check your version.

Why is submission crashing when opening a log file?

Try hardcoding a path on your local disk to **/tmp/** or another folder. Most likely you are trying to write to a file in your Current Working Directory which may be different when launch directly by Unreal.

Revision 12

I refactored my code and got a different score, what's up with that?

In testing we found that if you provide exactly the same responses to simulation frames (even between programming languages) you will get the same exact score. If your code has any built-in randomness to it (using rand(), a hiesen-bug) then you will not get a stable score when you are graded. In short, the seeds are fixed for the whole competition, and everyone is graded against the same set of seeds at the same simulation rate.

Is the version of the game in the Local Package different than the one used for Grading?

The grading and local environments use exactly the same version of the game.

When I run at 1x sim speed I get 1800 frames but at 4x I get 450. What's up with that?

Yes, this is a natural consequence of some decisions we made last week to make the game deterministic when graded on Kattis. We force the game to slice into "60 frames per second" (i.e. a fixed frame rate) and this means as we increase the frame rate, the numbers of frames process proportionally goes done. So the game is ticking with a greater "delta time".