**ESA: Computer In A Room Challenge 3 (CIARC3)**

**Report PART 4: User Console & Evaluation Phase**

**Team name:** Space Transformers

**Team members:**

ANAGNOSTOU NIKOLAOS

LAPPAS NIKOLAOS

MAKRAS ILIAS

MAVROMATIS MARIOS IOANNIS

VITALIS PETROS

Summary Report: Submit a 2–3-page report that includes:

• The logic behind your solution

1)

2)

**MAP CAPTURATION (our Commnader)**

We prioritized heavily into this objective and we gave it special value. For this objective we ‘ve built a smart and strong algorithm that controls the Melvin so that it maximizes efficiency.

Specifically we invented a python class called BitMatrix . This class creates a binary map that utilizes the fact that each memory slot is made out of 8 bits so we save up space of RAM . Using this important tool unlocked us the capability to make Melvin behave like a real person. Getting into more details , this map gives the Melvin the attribute of memory. This single element, allows the Melvin when taking pictures of the daily map , to update this memory and remember those places… But we hear you ask, why is this good? Well the answer to that is the pain of charge mode… This mode is inevitable but with this memory as far as Melvin goes and takes continuously more and more places of the earth , the Melvin acts smarter and the more it gets into charge when the Melvin travels long unnecessary distances … In that way the Melvin acts smarter and smarter and the pain of charging becomes lesser as the hours goes . **We are proud to say that this mission of the contest behaved perfectly and never failed us.** Its important to mention about how we approached the change in trajectories Melvin had to do. Lets begin from the “law” that Melvin had “when in a certain trajectory , first take photo of all the point of the trajectory and then change a trajectory”. Having that said, lets speak of how Melvin decided where to go next when a certain trajectory was accomplished. Here is a certain picture:

In this picture the spots which Melvin thought appears with red dots. But how it though I hear you ask… well what Melvin done is that he scanned in each spot all the 1.000.000 pixels around him and recalled his memory to identify how many pixels are vacant …or otherwise not captured.

Then, it does something really smart and that is that it selects the spot where the fellow criterial is fulfilled: commander takes the percentage of the map that is already photographed and then it compares it to the vacant/1.000.000 . If the vacant/1.000.000 is greater than the total percentage of undiscovered map then it sets it to target , calculates the correct velocities and Melvin is commanded to go to this target. And this logic made a huge difference on how fast commander actually takes photos of the whole map .

• How you prioritized the objectives

1. Beginning with the map
2. The first objective (and if same starting time, then choose the one that ends earlier)
3. Zoned objectives handled first over EB or secret
4. EB afterwards if no zoned objective in queue
5. EB waiting the first pings and if no pings and 1 minute passes, then Daily map for a specific time till the next ping
6. daily map along with secret objective scanning (image recognition algorithm running locally due to limited memory inside melvin)
7. if any errors occur, safety handler runs the daily map routine

• Resource management

1. Optimal orbits for less fuel consumption (vel\_calculation)
2. Often battery checks to make sure no battery runout occurs
3. Estimate when acquisition mode is needed (for picture capturing) and make sure it transitions to that mode with as much battery as possible (needed for the task)

• Functionalities implemented in the user console

1. Ηλίας …. (δεν ξέρω τι θα βάλουμε)

• Team performance assessment, including strengths and areas for improvement

1. Best implementation = Daily Map routine (Αναγνώ … )
2. Zoned objectives with enough area = Well structed and handled (like the first aurora)
3. Multithreading logic and automated submissions …
4. Zoned objectives improvements =
   1. handle better the smaller than lens areas, and take into consideration the coverage\_required …,
   2. implement the daily’s map routine, an array of bits, displaying the zoned objective area for optimal capturing technique
5. Beacon routine = when testing we were handling EBs with maximum distance from original point 150 maximum, (variety of orbits, more pings)
   1. needed more orbit changes in order to prune symmetric estimated points,
   2. ignore the area with radius 75 pixel (as center the first estimated point if wrong)
6. Beacon handling helped the daily map, cause while it did not wait for pings, it was capturing images
7. Forgot to implement the border of maximum speed while using narrow lens, and got API errors
8. Exceptions logger did not work with nohup, although we had tested it without it
9. Needed more memory support in order to run the sprite recognition algorithm inside MELVIN and reduce the amount of slots we used

• Diagrams and screenshots illustrating your solution

1. I have two concerning two EBs …
2. Need some from the console
3. Zoned/secret objectives

Code submission:

• Provide all code for both the console and MELVIN

• Ensure the code is well-organized **(using a git versioning tool is recommended)**

• Include **a README file** explaining the code structure and deployment instructions

**Your code and report will account for 60% of your final score in the challenge.**