Troubleshooting Tips And Techniques – AgentCubes Online

There is no one specific step-by-step approach to troubleshooting in general, nor in addressing problems with AgentCubes Online projects specifically. However, there are some general guidelines one can use in approaching problems that can well serve the troubleshooting process.

- **“What are you trying to do?”**
  - Begin with the question, “What are you trying to do?” It is important for the user to be able to explain what the intended behavior is, rather than simply describe what is happening that is not desired.

- **Read Agent Behavior**
  - Read the appropriate agent behavior; try to understand what it says before running the simulation. Computers do what they’re instructed, which may differ from what is desired. Computers read code, not minds!

- **Stop the simulation and reload the world before exploring.**
  - Start from “ground zero” when investigating a problem. Failing to do this is analogous to entering a theater half-way into the film: you’re likely to have missed some critical elements.

- **Check for stacked, missing, or duplicate agents**
  - If “mysterious” agents appear, check the contents of the world by removing one layer at a time to determine if there are hidden agents.
  - The single most common error in constructing worlds is to treat the world like a paint canvas and the agent placement tools (pencil and rectangle tools) like a paintbrush. The world is a layered structure and the placement tools will place agents on top of each other so that the agents underneath are typically not visible. Teach students to use the multi-agent (rectangle) tool for placing rows, columns, or blocks of agents, and the pencil tool for placing individual agents.
  - Agents will only run behaviors if they are on the world. Forgetting to add the controller agent to the world is a common mistake in PacMan, the behavior will not run if the agent is not in the current world

- **Try re-orienting the camera**
  - AgentCubes permits re-orienting the camera – what looks to the user like re-orienting the world. For example, one can turn the world on its side. Beginning students sometimes flip the world upside down or turn it 180 degrees, and forget to reload it before populating it with agents. There are some example projects in the troubleshooting exercises that show examples of this. The consequence is agent movement is backwards from what is expected, particularly with respect to using keys (e.g., arrow keys) to direct agent movement: the ‘up’ key moves the agent down or the ‘left’ key moves it to the right, etc. Unfortunately, the only resolution is to clear the world and re-build it.

- **Refresh the browser or clear the cache/cookies**
  - Sometimes the browser does not keep synchronized with the underlying elements (agents and world structure) of the project. Often this can be resolved by refreshing the browser window (not simply reloading the world).
- Sometimes, returning to the project screen will resolve the problem. Sometimes closing and restarting the browser may be required. Sometimes clearing the browser cache (via the browser options menu) may be required.

**Resolve error messages before running the project**
- AgentCubes Online produces several types of messages during agent behavior editing and world construction activities. Confirmation messages, such as when saving the world, are in green. Warning messages are in orange. These indicate a problem, but one that may not prevent running the project.
- A typical example is the addition of two methods with the same name. Note that, although the project may run, it will most likely not run as desired.
- Finally, error messages appear in red. These indicate problems with a rule’s conditions or actions. If the problem rule is reached, execution will stop. It is important to examine warning and error messages completely to determine the cause of the problem and correct it before attempting to run the project.

**Rule order can be critical:**
- Look for rules with no conditions. These rules will be performed unconditionally, which means any rules below such a rule will never be evaluated for execution. In general, rules that deal with “unusual” situations, such as game-ending rules, should be placed highest in the behavior list. As in life, first things first.

**Condition order can be critical.**
- Normally, since all conditions of a rule must be true, one might conclude that their order is not important, just as the order of selection of a sports team is not important; rather only the final team composition is important. But there are situations in which this is not the case in AgentCubes. The most illustrative example is one in which there is a timing condition and a probability condition. For example, if the conditions are:
  - Once every 1 second, with a 50% chance: This rule will be performed on the average about every 2 seconds.
  - However, reversing the conditions (with a 50% chance, once every 1 second), the rule will be performed every second.
  - The difference is due to how AgentCubes evaluates rules in the “While Running” method: every agent is scanned sequentially and then the cycle is repeated as fast as the computer can perform – hundreds of times per second or more. Since the 50% chance condition will occur about every other cycle, this means that the 50% chance will be true about every other pass through the agents. This is much more frequent than every second, so when the 1-second timer occurs, there is certain to be a 50% chance success within a fraction of a second of that event.

**Multiple ‘once every’ conditions can be tricky**
- Another twist regarding the ‘once every’ condition is the use of multiple rules with this condition in the While Running method. This requires a somewhat subtle understanding of how this condition is evaluated. Unlike some conditions which are relatively static, such as the shape of a nearby agent (for example, in the ‘see’ condition), the ‘once every’ condition is an event. When the event is evaluated in a rule, it is no longer true for evaluation in subsequent rules in the same simulation cycle. As a consequence, if there are two rules with this condition, and if the
condition is currently true, and if the first rule has a second condition which is not true, then the ‘once every’ event is no longer true when the second rule with the ‘once every’ condition is evaluated. An example of this is illustrated in the debugging exercises, Problem 07, the ‘Clock Problem’. This is a common problem for students with projects requiring, for example, alternating actions periodically. There are several ways to solve the problem.

- **Use conversational programming!**
  - This is a unique tool among software development environments and a very powerful tool, as well. Use conversational programming to determine which conditions are satisfied and whether associated rules will be performed as expected.

- **Examine agent attributes and simulation properties, as appropriate.**
  - Sometimes these values are not what you expect, were not saved, or have not yet been assigned.

- **Use the single step button.**
  - Watch behavior unfold incrementally, rather than trying to keep track of what happens in real time.

- **No need to play from the beginning to test one element of the game**
  - Avoid “playing the game” to examine a specific event. Rather, move agents into appropriate positions – that is, set up the test conditions – then run the simulation or use the single step feature.

- **For diffusion, check the diffusion formula carefully.**
  - Usually, a mis-entered formula will cause an error message and the input rejected, but not always. Sometimes the formula will pass syntax checking but not be correct semantically.

- **Hill-climb action should be “Von Neumann”**
  - Most designers use the hill-climb action to implement hill-climbing behavior. There are two forms of this action, one which allows movement in only four directions (referred to as Von Neumann neighborhood) and one which allows movement in eight directions (referred to as Moore neighborhood). Currently there are some situations in AgentCubes Online in which 8-way movement does not function properly. At this time using only four-way movement is recommended.

- **Carefully check hill climbing rules**
  - There are some situations in which implementing hill-climbing as a set of rules in a separate method is required. Examples include when one wants to limit movement to fewer than four directions or when actions in addition to movement (e.g., rotation) are needed. In such implementations, it is important to check the rules very carefully. As with the diffusion formula, the code may be syntactically correct but not semantically.

- **Check spelling of agent attributes and simulation properties.**
  - Note that these are created dynamically if not created deliberately. For example, if the user defines a simulation property named “total” (properly referenced as “@total”), but then mis-types the name as “@t0tal” or omits the “@” symbol, the simulation will run and dynamically create the additional name, but the behavior probably will not work as desired, because the rule will refer to a different variable.
  - It is easy to see the complete list of variables at any time that AgentCubes Online recognizes within a given project: in the actions palette, select the first operand field.
of either of the “set” actions; AgentCubes Online will display the complete list of variable names that have been defined for that project. This should be done when the project editing screen is first entered to observe the list of variables before the project runs. Then, after running the project and stopping it where the problem seems to occur, repeat the examination of the variable list. This will show any variables that were created dynamically via a “test” condition or “set” action, for example.

- **Simplify your world for debugging**
  - Remove agents not involved in the problem to permit examining other agents without interference. For example, if there are six agents pursuing a single agent, and the problem concerns whether hill-climbing is working properly, temporarily remove all but one of the tracking agents so that you can focus on one agent at a time.
  - Make a new world with only the agents involved in the problem. Sometimes, simply removing “extraneous” agents may not permit understanding the problem, especially if there are buried agents, or possibly a corrupted world. Starting “from scratch” with a simplified world can help address whether the problem is with the world construction or agent behavior.

- **Careful not to delete worlds when using ‘switch-to-world’ action**
  - Switching worlds: AgentCubes Online requires the designer to specify a destination world in the Switch-to-world action. The destination can only be an existing world. However, once the selection is made, if the world is subsequently deleted, the Switch-to-world action will point to a non-existent world. This will cause an error if the rule containing this action is executed.