RAMBOTS

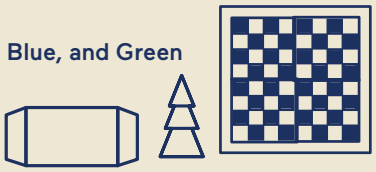
DESIGNED BY KORY HEATH

2-4 | MEDIUM | COMPLEX

EQUIPMENT

- N+1 trios of Red, Yellow, Blue, and Green
- Chessboard
- N Large Clear pyramids
- N Privacy Screens

N = Number of players



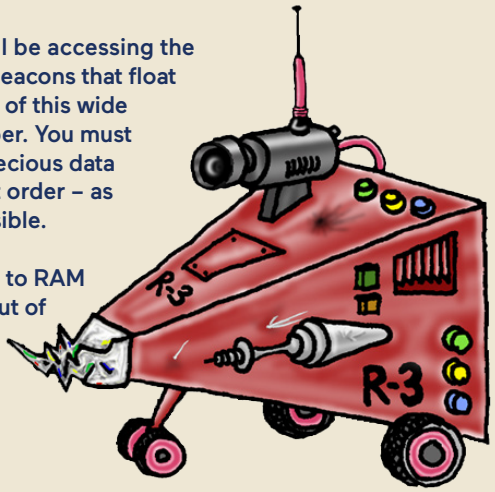
OVERVIEW

Deep in the heart of The Complex, you have been awakened by the Master Priority Scheduler. Yet again, you must perform your data collection tasks. No more time for wondering what lies beyond The Complex, or if "outside" really even exists. You now have a task.

And uh-oh, it's a Multi-Task.

Other 'bots will be accessing the glowing data beacons that float near the center of this wide metallic chamber. You must collect your precious data – in the correct order – as quickly as possible.

Do not hesitate to RAM the electrons out of anyone who gets in your way!



SETUP

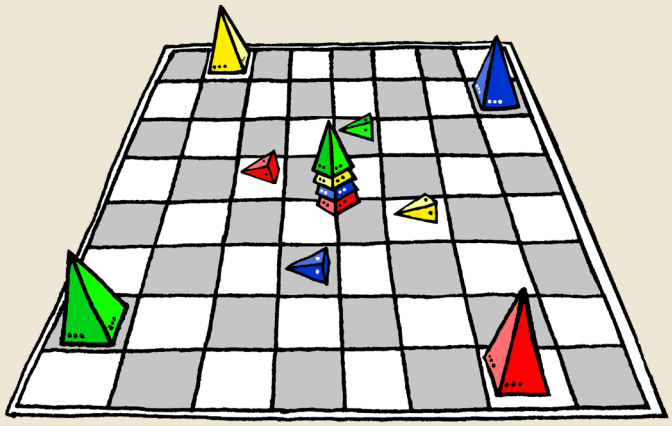
STEP 1 - DISTRIBUTE CODE POOLS: Stack up all of the pyramids into monochrome trees and place a full set in front of each player. Players will use these pieces to create instructions for their RAMbot. Each player also receives a Clear piece (called a Compilation Cap) along with a privacy screen to hide their programs behind. Ideally, the privacy screens will be color-coded to display the color assigned to each player.

STEP 2 - POPULATE THE BOARD: Place the extra set of trios on the gameboard and sort those pieces by size. The Larges will become the RAMbots, the Smalls will become the Data Beacons, and the Mediums will become the Precedence Stack.

Place the RAMbots in the space shown below, closest to their assigned players. (If there are fewer than 4 players, set aside the unused RAMbots.)

Place the Data Beacons, lying on their sides (direction doesn't matter) in the spots shown below. (For a more unpredicable setup, place the pieces in the indicated spots but in random color order.)

Create the Precedence Stack with an initial random order and place it in the middle of the four center squares.



STEP 3 - CREATE GOAL STACKS: Each player now chooses four pieces from their Code Pool (one of each color but whichever sizes they choose) to create the Goal Stack for the player to their left. These stacks will determine the order in which the Data Beacons must be tagged (first on top). Don't let anyone see which pieces you've chosen until all players are ready to reveal at one. Place the Goal Stacks in the corners of the gameboard, not on the corner spaces themselves, but on the edges.

CONT. →

RAMBOTS (CONT.)

GOAL

Program your RAMbot to tag all four Data Beacons, each of which has first been activated, in the order indicated by your Goal Stack, before anyone else!

WHO GOES FIRST

This game doesn't have a traditional turn order. Everyone does their programming simultaneously, then everyone's programs are run at the same time, with the order of multi-tasking operations being dependant on both the Precedence Stack and the piece sizes used in programming.

HOW TO PLAY

During each Programming Phase, you will choose a sequence of five commands for your RAMbot. Then, during the Execution Phase, you will hope for the best as your commands are carried out!

THE PROGRAMMING PHASE

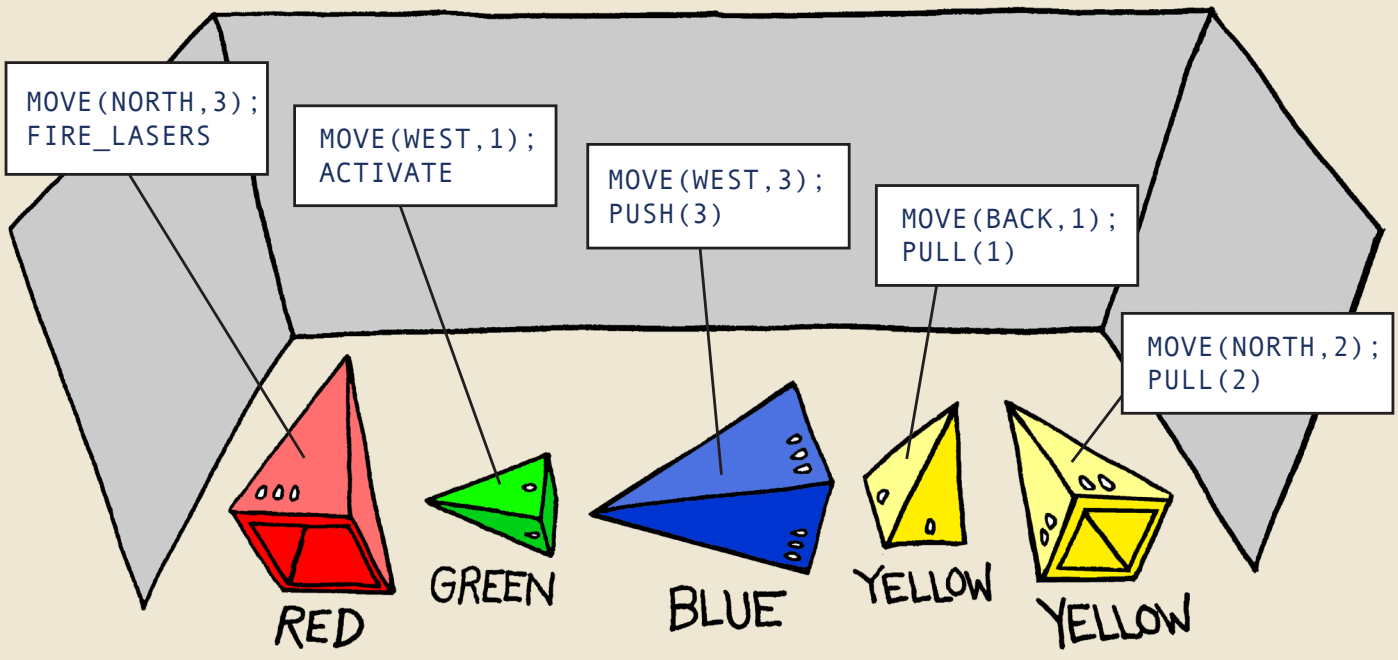
At the beginning of this phase, all players set up their screens and begin "programming" their RAMbots. You may lay out up to five of your code-pool pieces behind your screen; they will be executed in order from left to right.

Each instruction piece represents a single RAMbot action, which will cause your RAMbot to move and then shoot a beam of colored energy. Each color has a different effect. Blue and Yellow beams Push and Pull, Green Activates (i.e. causes to stand up) a piece, and Red does damage.

The maximum number of instructions is five, but fewer is allowed.

When all players have finished programming, everyone lifts their screens, and the game moves on to the Execution Phase.

A TYPICAL PROGRAM



RAMBOTS (CONT.)

THE EXECUTION PHASE

To begin this phase, each player should slide their leftmost instruction forward to the center of the nearest board edge, an action known as "loading the program register." One of these instructions is about to be executed; to determine which one, first look at the sizes of the pieces. Smaller pieces run more quickly than larger pieces, so smaller pieces always take precedence over larger ones. If there is a single smallest piece, simply execute that instruction.

If there is a tie for smallest piece, compare the colors of the tying pieces to the Precedence Stack. Higher colors always take precedence over lower ones. If there is a single highest color among the tying pieces, simply execute that instruction.

If there is another tie, then the instruction pieces in question must be identical. In this case, compare the colors of the appropriate player's RAMbots to the colors in the Precedence Stack. The player whose color is on top executes first. (This is easy to see, assuming you've arranged the precedence stack according to the initial RAMbot colors. The player whose color is currently on top of the precedence stack will have the highest precedence during a tie, and so on clockwise around the table.)

After an instruction has been executed, the instruction piece should be returned to its owner's Code Pool, and that player's next leftmost instruction should immediately be pushed forward into his or her program register. Once again, follow the above rules to determine which instruction should be executed next, and then execute it. (Note that it is perfectly possible for a player to execute two or more instructions in a row.) Repeat this process until every instruction has been executed.

After executing all instructions, move the topmost piece of the Precedence Stack to the bottom, then the next Programming Phase begins.

EXECUTING INDIVIDUAL INSTRUCTIONS

Each instruction in your program will cause your RAMbot to move and then fire a beam of colored energy in front of itself. Each piece in a program should either be lying down and pointing in one of the four cardinal directions, or standing upright on its base. The orientation of the instruction piece indicates how your RAMbot will move, and the color of the instruction piece indicates what kind of beam your RAMbot will fire.

MOVEMENT

An instruction piece lying on its side tells your RAMbot to move in the direction the instruction piece is pointing. When you execute this instruction, move your RAMbot one, two, or three spaces in the appropriate direction, depending on whether the instruction piece is small, medium, or large. If your RAMbot is not already facing in the direction that your instruction piece is pointing, the first unit of the movement instruction will be used to reorient your RAMbot in the appropriate direction. So, for instance, if you have a large movement instruction piece pointing in a different direction than your RAMbot, you must first reorient your RAMbot (using one unit of the movement action), and then move it two spaces in the appropriate direction. A small-sized movement instruction will therefore reorient your RAMbot without moving it if it's not already facing in the appropriate direction. If at any time your RAMbot is standing on its base (as it is at the beginning of the game), the first unit of a movement instruction will tip your RAMbot down and point it in the appropriate direction, to be followed by the rest of the movement action.

An instruction piece standing on its base represents "reverse gear"—it causes your RAMbot to move backwards away from the direction that it's currently pointing, for one, two, or three spaces. If your RAMbot is currently standing on its base, an upright instruction will not move your RAMbot at all.

CONT. →

RAMBOTS (CONT.)

PUSHING AND RAMING

If, during one of your movement instructions, your RAMbot moves forward into or backs into a space that contains a beacon or another RAMbot, the object will be pushed. If there are objects directly on the other side of the pushed object, they will be pushed along with it. If an object cannot be pushed any further (because it's against a wall, or it's against objects which are against a wall), your RAMbot simply stays where it is (though you do still make contact with the object). If your RAMbot runs into a wall, nothing special happens.

If the nose of your RAMbot hits another piece, you have RAMed that object. (Backing into an object, or getting pushed into an object during someone else's instruction, you do not RAM it.) Multiple hits during a single instruction counts as one RAM. Whenever you RAM another player's RAMbot, you cause damage to that RAMbot. Take the highest precedence piece from that player's Code Pool and add it to your own Code Pool. In other words, take the smallest piece available in that player's Code Pool; if there's a tie for smallest piece, take a piece of the color that's highest on the Precedence Stack. If there are currently no pieces in that player's Code Pool, you don't get to steal any pieces. You may not steal pieces from an opponent's currently running program.

If you RAM an upright object (whether it's a beacon or a RAMbot), tip it down onto its side, facing away from the point of impact. If that object matches the color currently on top of your goal stack, you have tagged a goal. Remove the top piece from your Goal Stack, and add it to your own Code Pool.

ENERGY BEAMS COLORS

After your RAMbot moves, it will fire a colored energy beam in a straight line out in front of itself. The beam's color is determined by the color of your instruction piece. The beam will affect the first object it hits. If a beam hits a wall, or fires straight upwards, it has no effect.

BLUE-PUSH: A blue beam will push any object it hits (along with anything else that the object runs into) away from your RAMbot for one, two, or three spaces, depending on the size of your blue instruction piece.

YELLOW-PULL: A yellow beam will pull any object it hits toward your RAMbot for one, two, or three spaces, depending on the size of your instruction piece. If an object is pulled all the way into the nose of your RAMbot, the object remains in the space next to your RAMbot. This does not count as a RAM; you cannot damage another player, tag a goal, or knock over an upright object in this fashion.

RED-DAMAGE: A Red beam will damage any RAMbot that it hits. (This is in addition to any RAMing damage caused during this instruction.) Take the highest precedence piece from that player's Code Pool and add it to your own. (The Red beam doesn't actually RAM and doesn't knock over the target piece.)

GREEN-ACTIVATE: A green beam will set any object it hits upright.

HOW TO WIN

The first player to tag all four goal colors in the order specified by their Goal Stack is the winner!

OTHER NOTES

Compilation Caps: These are used to indicate who's finished programming (so you can tell when everyone's ready). Drop your Clear onto your Goal Stack to indicate that you're ready to run. (You may also remove the Cap to "uncompile" and change your program if others are still coding.)

Code Pool Disclosure: Although you may hide your work while Programming, your opponents do get to know what pieces you are working with. Therefore, you must give honest answers to any questions you are asked about the number, sizes, and colors of pieces in your Code Pool.