- **Personal information:**

  Name: Gabriel Bercaru

  E-mail address: gabibercaru@yahoo.com

  IRC username: Gabriel (Also kept it Gabriel_)

  Background info: I am a first year student at the Faculty of Automatic Control and Computer Science, Polytechnic University of Bucharest, Romania. So far, I have gathered experience in the following fields: General Programming (mostly studied C language, C++, bash scripting and Matlab), Operating Systems, Data Structures, Algebra and Numerical Methods. During these courses I have also been involved in several team projects.

- **Project information:**

  Project title: “Add exec option to search”

  Project summary: “BRL-CAD is a solid modeling computer-aided design system.” Currently, the user can look for the objects in a database by searching after a particular pattern. Using the “search” command, the user can retrieve all the objects inside a database matching the given criteria. However, it would be desirable to be able to apply a set of commands on found objects. This is what the “-exec” option aims to do. It it similar to the “-exec” commands applied to “find” on UNIX systems. Such an implementation for BRL-CAD “search” command would be a time-saving option, since the user can automatically apply a set of commands on the found object in a single line. This new option is thought to be implemented as a general mechanism, so it must link directly the “search” command results with all the commands which can be applied to an object in the database.

- **Implementation details:**

  I think that this projects mainly consists of two parts:

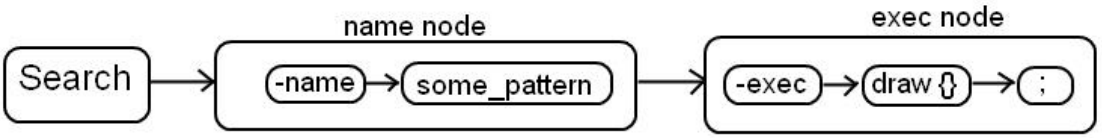
1. Addind the “-exec” tag between the “search” command options and implementing the function that parses this new command argument.
2. Defining the “-exec” general syntax and then processing the words passed to this new argument so we can execute the requested commands accordingly.

Given the already implemented command line arguments processing method, we need to follow it and insert a new case for the “-exec” argument. We determine the type of this node (“-exec”), create the suitable node type and then add this plan node at the end of the existing plan, just like it is currently done for the other nodes. So, the linked list will have one more node type. This is related to the first part of the project.

As for the second part, I've thought that the newly added “exec” node could represent itself another linked list. Let us take the basic example: drawing the objects that match the user-inputted pattern would be achieved with the following command:

```
search -name some_pattern -exec draw {} ;
```

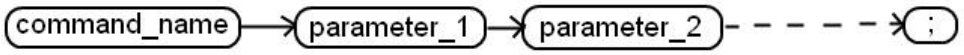
Parsing this command would result into this:



Basically, the exec node would look like this:

```
typedef struct commandNode {
        char *stringToProcess;
        struct commandNode *nextString;
}command;
```

The “stringToProcess” variable will hold the entire command we are currently working on. By separating it into tokens we will eventually be able to find what command should be called and what parameters should be applied to it.
In fact, I believe that every command passed to “-exec” could end up being another linked lists, such as:

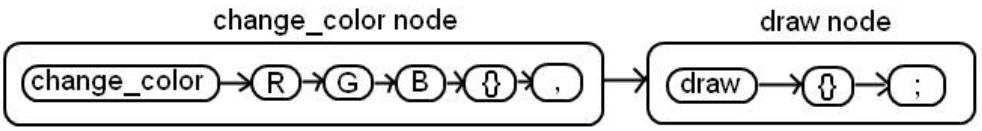


NOTE: the names used to illustrate this example are just orientative, they will be changed to conform with the rest of already implemented data types/structures/nodes.

As I have understood from the conversations held on the IRC channel, “-exec” should be able to perform multiple operations at once on the same search results. The user could, for instance, change the colors of the found items and after that, draw them:

search . -name some_pattern -exec color_change R G B {} , draw {} ;

Below is a possible interpretation of this command:



One thing that remains to be decided is the general syntax. There should be a command separating character, such as an IFS (I used a comma ',' in the above example, but this can

be changed if needed). We also need to decide how will “-exec” know which commands to execute. There are two possiblities:

1) Assume that “-exec” must be called the last one and no other chained commands occur after “-exec” (so something like “search b* -exec change_color R G B {} , draw {} ; ls” would not be possible)

2) Add a pair of separators (for example '<' and '>') which should enclose the commands which are called by “-exec” (“search b* -exec <change_color R G B {}, draw {}>; ls” might be possible).

Also, from the discussions held on the IRC channel, I have understood that the keywords map (the keywords which trigger function calls) are currently inside MGED. However, if we want to implement a feature such as “-exec” inside “search”, we would have to bring the keywords list at a lower level, perhaps a new core library which is shared by both MGED and libged(since search's functionality is coded inside libged's “ged_search()” function).

- **Project deliverables:**

May 23 (coding begins): I will start the work by trying to get (at first) a working code of the basic example, such as executing a “draw” command on objects found by the “search”. After testing this particular case functionality I will pass on to generalization and automating the process. This would involve modifying MGED and libged so the table mapping commands to function calls would be accessible to the commands passed to “-exec”.

June 20 to June 27 (mid-term evaluation): for the mid-term evaluation, I intend to have a working version of the “-exec single command”, tested and documented.

August 15 to August 23 (final-term evaluation): have the “-exec multiple commands” working, necessary tests for it and also documentation.

- **Development schedule:**

Until May 23 (community bonding period): familiarize myself with SVN (as I am currently used only to Git), BRL-CAD's commands and read the documentation in detail, communicate on the IRC channel, discuss some other implementation details and ask for help if I need it. Also, I will make sure to have a fully working BRL-CAD environment, on both Windows and Linux.

One important aspect I have to mention is that from May 25 to June 16 I will not be able to fulfill the minimum of 8 hours/day due to my summer exam session. However, I will be able to work on the project for about 15 hours/week (or maybe even more, depending on how much time I need to allocate for studying for the exams). Excepting this period, I will be able to work on the project for about 50 hours/week or possibly more (to compensate my absence in the first part).

Weeks 1 – 4 (May 23 – June 19):

Create the “demo version” of “-exec draw {};”
Migrate the functionality from MGED to a new core library.
Implement the “single command” version of “-exec”.

Test and document the new search functionality.
Week 5 (June 20 – June 27): Mid-term evaluation
Weeks 6 - 12 (June 28 – August 14):
Generalize the single command version, test it thoroughly.
Review the code, add more comments where necessary and write the documentation.
Week 13 (August 15 – August 23): Final-term evaluation

- **Why BRL-CAD?**

Because of my interest in computer graphics and animations and the wish to contribute to one of my favourite computer-related fields.

Below is a link to my patch, where I tryed to implement a new functionality to the “search” command: display how many items have been found. I understood why my approach would not be suitable but unfortunately, I did not have enough time to try to implement what Sean Morrison suggested.

https://sourceforge.net/p/brlcad/patches/433/

- **Why me?**

I am not the best programmer, nor the worst one. However, I believe that my lack of experience in developing such big projects or maybe even lack of knowledge in some cases is compensated by enthusiasm, willing to learn and perseverence. Since I have no other commitments planned for the summer, I can dedicate all my time to this project.

Thanks for reviwing my proposal!