Spring 5-5-2015

Motion in the Field: A Study of Movement in Computer Science

Catherine Martin
Syracuse University

Motion in the Field: A Study of Movement in Computer Science

A Capstone Project Submitted in Partial Fulfillment of the Requirements of the Renée Crown University Honors Program at Syracuse University

Catherine Martin

Candidate for Bachelor of Science in Applied Mathematics and Computer Science and Renée Crown University Honors

May 2015

Honors Capstone Project in Computer Science

Capstone Project Advisor: _____
Jae Oh, Professor

Capstone Project Reader: _____
Edmund Yu, Professor

Honors Director: _____
Stephen Kuusisto, Director

Date: 5 May 2015

Abstract

In this project, I use a system that combines the use of a motion controller to obtain information about physical motion of a user with the transmission of outputs to a quadcopter that interacts with the physical environment. In the manual mode, the user sends commands to the quadcopter using hand controls. I utilized an existing system but increased usability to prevent accidental flight and created commands that are more intuitive for the user. In the automatic mode, the user gives a single command to the motion controller and it runs the A* path planning algorithm over a representation of a known environment to make the quadcopter find and execute the shortest path to its goal.

Executive Summary

In this project, I attempt to explore the concept of physical human-computer interaction through the use of two technologies: the Leap Motion Controller and the Crazyflie Nano Quadcopter. I did this by first working on a manual system for flying the quadcopter via the Leap. In this part of the system, the user uses his hands to give commands to the Leap Motion, which the computer then translates into commands for the Crazyflie. Next, I explored the use of the quadcopter for intelligent path planning within the known environment. In this section, the user gives a representation of the world where he wants the Crazyflie to operate, a start destination, and an end destination, and the Crazyflie autonomously goes from the start to the end via the shortest path.

To implement the manual flight portion, I utilized a system that already existed as a basis and fixed its problems with usability. The company that created the Crazyflie had already built a system that would use the Leap Motion Controller to fly it, and they posted the code online. I downloaded this code and made it more user friendly by reducing accidental flights and making the commands easier for a new user

to learn and execute. This involved learning the basics of how the Leap interpreted data and how the Crazyflie receives commands and then a fair amount of experimentation to determine how the commands best matched the desired flight. Once the basic commands were determined, however, they were easy to manipulate and predict going forward.

To implement the automatic flight portion, I created a program that would take in a representation of the world as a grid of one by one squares and execute a shortest path finding algorithm, called A*, over it. The algorithm is considered an intelligent algorithm because it changes its idea of where the shortest path is based on new information during execution. To make this program, I had to learn the details of how the algorithm worked, and then it was just a matter of writing in out in the way that I needed for the information that I wanted to give and receive. After that, I changed the quadcopter flight program to allow it to take the values that my program generated and used them to execute the shortest path in the physical environment. This also involved a fair amount of experimentation as I needed to figure out how to make the quadcopter move one foot in each direction, but otherwise the implementation was just a matter of writing how I wanted the system to interpret the commands.

Although this system operates on a very small scale, I used it as a model for larger systems. Technologies that can physically interact with humans are becoming more common and have very big implications. Motion sensors can be used to quickly and accurately take commands which can be useful in time sensitive systems. Quadcopters have even greater consequences, as they have been used for a variety of things, from package delivery to drone attacks in war zones. The Leap Motion Controller and the Crazyflie Nano Quadcopter are such small representations of what their respective technologies can accomplish that it makes the fact that they can do so much incredibly impressive, and I hope to show that with this project.

1. Introduction

As personal computers become faster, cheaper, and more ubiquitous, the focus of developments in computing has been shifting. New arenas have developed to allow people to start making new and innovative technology that will allow for an entirely different computing experience. It is becoming more common that these technologies are designed to change the way the user interacts with the physical world. From Google Glass and the Microsoft HoloLens, which turn the world into a screen, to control systems that fly airplanes, to diagnostic tools that are revolutionizing medicine, technology is being

reintroduced as a way to change everyday life. Tracking and control systems like motion sensors and quadcopters are just a small, but important, part of it.

Motion sensors have come a long way in recent decades. Originally only used to determine whether or not movement was present, they have steadily become more accurate and have found many more applications within computing. Tracking motion sensors have most famously been used for gaming since the release of the Microsoft Kinect for the XBox 360, but the Kinect as well as other motion controllers have been introduced to developers for use, which has changed their place in computing.

Similarly, computer controlled systems like quadcopters are no longer the toys that they used to be. They now carry the accuracy and range to be able to do a myriad of tasks, from filming rarely viewed angles with mounted cameras to delivering packages via Amazon to carrying out acts of war. These systems can be incredibly powerful and have large implications in the physical world. Despite the fact that the quadcopter used in this project doesn't have anywhere near that kind of power, it is still amazingly agile and responsive considering its small size.

This project focuses on the use of the Leap Motion Controller, released in 2012, for its applications as a controller for a mini quadcopter, called the Crazyflie Nano Quadcopter. The Leap Motion Controller in a highly accurate motion controller that tracks the movements of the hands and fingers, with limited views of the attached arms. This paper focuses on the process of setting the Leap Motion Controller up for use with the Crazyflie Nano Quadcopter, as well as different features needed in software to create a successful program, and the process of implementing an A* path planning program for autonomous flight. It is organized into several different sections. Sections 2 and 3 will introduce the hardware and software components, respectively, that are utilized in the system. Section 4 will discuss the previous system that was used as a basis for the manual flight system and its flaws. Section 5 explores the improvements made to the manual system, while section 6 introduces the A* algorithm and discusses its implementation in the automatic system. Section 7 will explore future work to be done in the system, and is followed by the conclusion in section 8.

2. Hardware

There are three main hardware components at use in this system. First, the Leap Motion Controller is a motion detector designed to track the movements of the hand as it moves through space at 290 frames per second. It has a 150 degree field of view that extends between 1 inch and two feet above the controller,

and tracks movements within 1/100th of a millimeter. The controller uses a Cartesian coordinate system, where, when seen from the top, the X axis extends to the left, the Z axis extends forward, and the Y axis extends directly up from the center of the controller, as shown in Figure 1. It is designed to provide specific details about the number of hands in frame, whether the hand is a left or right hand, the direction of the fingers, etc. If it is unable to find the necessary information about the hand that it has been tracking, it uses the visible parts of the hand against its internal model as well as its past observations to predict the positions. The Leap can also detect gestures or general geometric motions. These characteristics are utilized to create the controls for the Crazyflie Nano Quadcopter.[1]

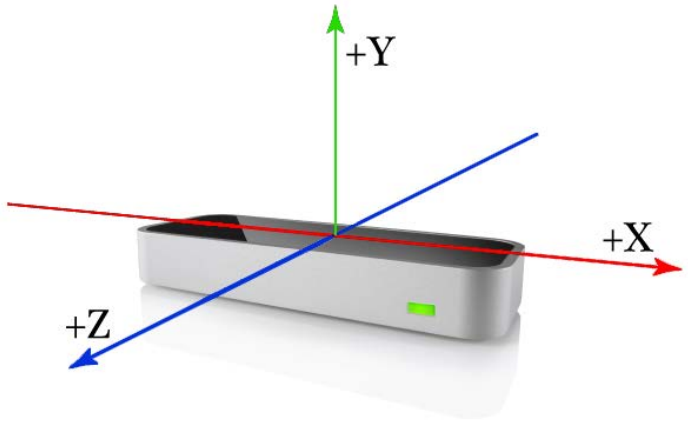

Figure 1. Photograph of the Leap Motion Controller with visual representations of the directions of the axes in the field of motion above it.

Second, the Crazyflie Nano Quadcopter is a programmable mini quadcopter. It is roughly 4x3x1 inches, weighs 0.3 ounces, and flies with six degrees of freedom, which allows the user to control the roll, pitch, and yaw, along with the thrust. The flight time on one charge is approximately 7 minutes depending on activity. It communicates with the computer using a 2.4 GHz USB radio dongle called the CrazyRadio and has between a 250 Kbps - 2 Mbps communication data rate. The quadcopter is initially programmed to fly using a computer manually, with a Playstation 3 or XBox 360 controller being recommended. It is designed for quick, agile flight.[2]

Figure 2. Photograph of the Crazyflie Nano Quadcopter, showing the frame, battery, and set of four blades. [3]

Finally, the system runs through a computer, which needs to meet a few technical standards, including a Windows 7 or 8 (if using a Windows machine) operating system, at least 2 gigabytes of RAM, and two USB 2.0 ports.[2] The one used for this project is a Lenovo Thinkpad, which meets or exceeds all of these requirements.

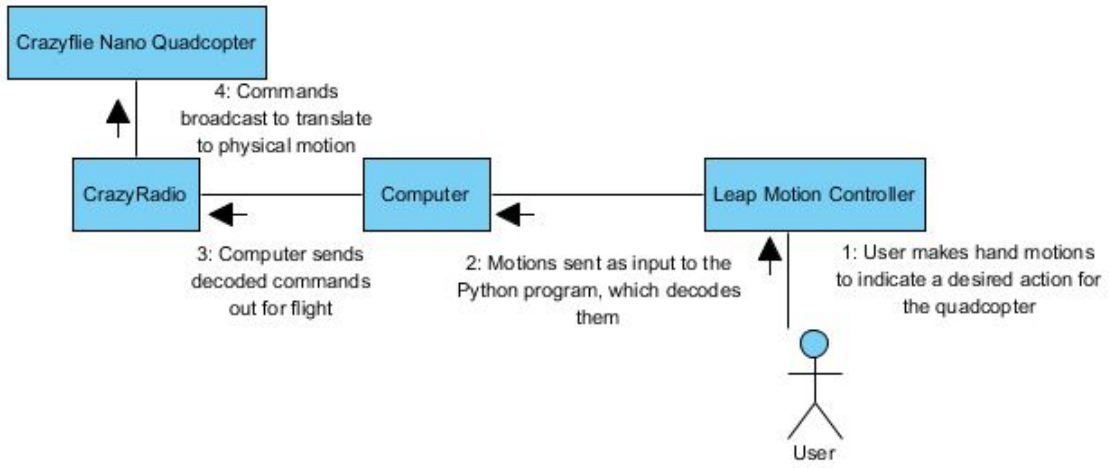

Figure 3. Hardware interaction to show how the user interacts with the Leap Motion Controller to operate the Crazyflie Nano Quadcopter.

As shown in Figure 3, the user interacts directly with the Leap Motion Controller. The user sends commands through gestures, which the Leap Motion relays to the computer. The computer uses the

Python program to decode the movements into physical commands, which are relayed to the Crazyflie through the CrazyRadio.

3. Software

Many different types of software had to be used to be able to interface the Leap Motion Controller and the Crazyflie Nano Quadcopter. Before the Crazyflie can be used, the computer from which it is being flown must have a few different packages installed. PyUSB 1.X needs to be installed to be able to communicate with the Crazyradio dongle, and libusb is needed as the back end for PyUSB. It also requires PyQT4 for development with the QT4 framework. Although a controller is not used for this project, the Crazyflie package requires the installation of pyGame, which is normally used to get readings from USB controllers, before it will run. Furthermore, to be able to develop with the Crazyflie, Python 2.7 is needed and can be run from the terminal.[4] The interaction between these components is shown in Figure 4.

Once these packages are installed, the Crazyflie Nano Quadcopter runs through an interface system with a graphical user interface (GUI) that the company that makes the Crazyflie, Bitcraze, designed. It allows the user to connect to and communicate with the Crazyradio dongle, as well as track the Crazyflie's battery level, roll, pitch, yaw, thrust, and level. Bitcraze also allows the user to program the Crazyflie by editing the files that make up this GUI, referred to in documentation as the cfclient.

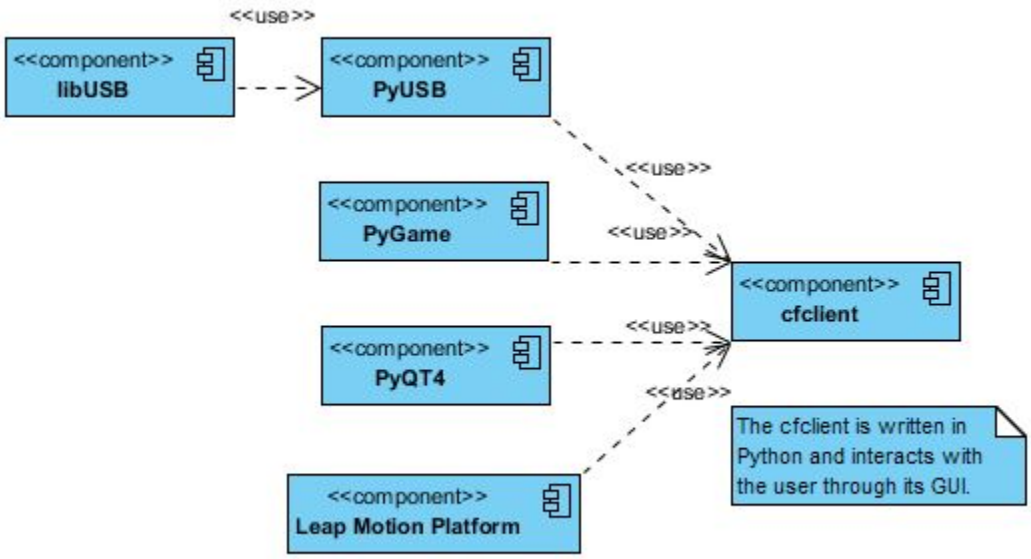

Figure 4. Software interaction between the necessary software packages, the Leap Motion software, and the Crazyflie cfclient that allows for the control of the Crazyflie through the Leap.

For the Leap Motion Controller, the only software that needs installation is a package produced by the company that gives access to development tools as well as applications that the company and other users have created.[5] During development, it does not require any external software to run simple programs, but is also scripted using Python 2.7, though other options are available.

4. Previous Work

An important part of the software development process is researching previous work to find out what others have done to be able to learn from and expand upon it. Although the original intention of this project was to integrate the Leap Motion Controller and the Crazyflie Nano Quadcopter, it was discovered that the company that designed the Crazyflie, Bitcraze, had already interfaced it with the Leap Motion Controller, and had the package that they created available for development. This package was downloaded and accessed via the Bitcraze Github repository, and was used as the basis for the rest of the project, taking into account the need to update the statements for use with Python 2.7 and the Leap Motion Controller. Instead of recreating an existing system, the focus of the project shifted to improvements on the existing system.

The system that the company developed used any one hand that was in frame, and directly tracked its roll, pitch, yaw, and thrust based on hand position. The Leap Motion Controller can track movements up to 1/100th of a millimeter, so any normal movement of the hand that would occur from involuntary muscle movements or not holding the hand perfectly straight or flat would change how the Crazyflie flies. Also, the thrust was purely based on the hand's position on the Y axis, so it could gain or lose altitude very easily without intention. Also, due to having a system based on the presence of only one hand and its position in frame, it is easy to fly the Crazyflie accidentally, which could lead to crashes.

This system left room for several improvements:

- Stricter requirements for the user before the quadcopter flies to ensure intentionality
- Commands that give the user more control while maintaining intuitiveness
- Intelligent design within the system to allow for path planning input

There are many other improvements that could be made on this system, but these are out of the scope of this project. They will be discussed future work in section 7.

5. Interaction Design

An important part of software design is the usability of the system. Typically, systems that are difficult to use or are counterintuitive to the majority of users often don't get used and fail to competing systems that users find more natural. The two main flaws in the usability of the existing Leap Motion and Crazyflie interaction were the ease of accidental flight and the difficulty of control. To decrease accidental flights, I implemented a two handed system that forces the user to show intentionality before interacting with the system. To make the controls more natural, I created commands that are similar to the movement that the quadcopter would make in response and made it less sensitive to natural inadvertent human movement.

To make sure that the user intends to use the Crazyflie Nano Quadcopter before it begins to fly, the new program necessitates that two hands be in frame, that one be a left hand and one be a right, and that the hands be in a certain orientation. If the left hand is closed, then the Crazyflie does not move. If the left hand is flat, however, then the Crazyflie's actions are determined by the position of the right hand. Whether or not the hand is open is determined by the independent direction of the pointer, middle, ring, and pinky fingers to ensure without a doubt that the hand is intended to be open before flight. This is the safer way to proceed to ensure that the user does not unintentionally begin to use the Crazyflie.

The next problem was determining user controls that would be intuitive and purposeful for the user and safe for the quadcopter. Once the program determines that the left hand is open, it determines the Crazyflie's actions based on the position of the right hand, as listed in Figure 5.

Position	Command
Flat	Hover steadily
Fingers below the line of the wrist	Fly forward
Fingers significantly above the wrist	Fly backward
Horizontal left tilt	Fly left
Horizontal right tilt	Fly right

Figure 5. List of commands that the right hand gives when accompanied by an open left hand during manual flight.

This model was implemented because it is very intuitive, and therefore very easily picked up by a new user. Furthermore, these commands exist within thresholds of roughly 50 degrees. Therefore, if the hand is relatively flat, for example, then the Crazyflie will remain straight. The user has to show intention to

move the hand for it to exhibit the matching behavior. Then the output changes within intervals that vary greatly for the computer, but seem to be relatively small differences to the user. The differences in tilt can therefore allow the user to command no tilt, left, right, forward, and back, with the left/right and forward/back commands able to occur simultaneously.

To determine thrust, the program uses the difference in elevation between the left and the right hand, once again using thresholds to determine intentionality. If the hands are relatively even, then the Crazyflie will hover steadily. If the right hand is lower than the left, it will slowly descend; if the right hand is higher, the Crazyflie will slowly rise. This is also intuitive due to the fact that the only hand that the user has to move throughout the input process should be the right hand. This therefore makes is easy for the user to learn and remember which hand determines the thrust.

The system is designed to be able to be quickly learned by a user, while still allowing for human error and imperfection. This makes the Crazyflie easier to control, which allows the user more ability to determine where and how it flies.

6. Intelligent Path Planning

An important feature of controlled systems is their ability to interact with the physical world. This includes the ability to get from a starting point to an ending point efficiently and without crashing. Unfortunately, the Crazyflie Nano Quadcopter doesn't have any sensors on it for receiving information about its environment. Instead, a system was implemented using the A* path planning algorithm with input of a known environment and output of static commands to allow the Crazyflie to perform a sequence of commands intelligently within the environment, despite its inability to interact directly with the scene. Changes in environment are recorded by manually changing the program.

The A* path planning algorithm is a heuristic artificial intelligence algorithm for finding the shortest path from a given starting point to a given ending point. It does this by using the value of a function, f(n), which is the estimation of the shortest path, defined by $f(n) = h(n) + g(n)$. The function g(n) is the exact calculated cost from the starting point to the current state in the environment. The function h(n) is the estimated cost from the current state to the ending point, assuming that there are no obstacles between. In this implementation, this is estimated using the Manhattan method, which takes the straight line distance without diagonals. This number is often incorrect, but it allows the system to make an educated guess for where the best direction to move is. The path is then navigated from the starting point to the end point by

investigating the state with the lowest f(n) value at each step. Then, once the ending point is reached, the value of f(n) becomes g(n), and the shortest path is established.[6]

This algorithm was implemented with two main parts. The initial part was creating an offline A* program to generate the quadcopter the commands that it would need to be able to reach its destination. The initial environment is passed into this program as a matrix, representing one foot by one foot blocks of space in the room based on whether or not they contain an obstacle. As shown in Figure 6, the matrix is based on binary values, with a zero representing a viable space and a one representing an obstacle. The path is constructed by searching neighboring entries within the matrix and updating the shortest path that it takes to reach each state. It explores the states with the shortest f(n) values until it has reached the goal state. Once the shortest path is established, it is followed backwards from child to parent to construct an array of directions (forward, right, etc.) to be passed to the quadcopter, indicating the path that it should follow in the real world environment. This array is then passed to the program that gives the Crazyflie direct instructions so that it can execute the following of this path. Figure 7 shows an example of a list of nodes that have been explored in search of the shortest path. Each node is represented as a list that carries the values of the node's f(n) value, g(n) value, h(n) value, parent node, and status as an explored node. The path list contains the keywords "start" and "end" to let the quadcopter know when to rise and land, and in between carries all of the commands to reach the shortest path. In this example, it is assumed that the start is the top lefthand corner, the goal is the bottom righthand corner, and the quadcopter is facing right.

```
environment = [
    [0, 1, 0, 0, 0],
    [0, 0, 0, 0, 0],
    [0, 0, 1, 0, 0],
    [0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0]
]
```

Figure 6. An example of an environment matrix given to the A* program. The "0" represents an open space and a "1" represents an obstacle.

Figure 7. An example of a list of explored nodes with their f(n), g(n), h(n), parent, and explored status, and the resulting shortest path that is passed to the quadcopter flight program.

The second part of the A* path following was setting up the Crazyflie to be able to follow the commands given in the aforementioned array. The system will still allow for the user to manually fly the quadcopter with the Leap Motion Controller, so first the user needs to make the specified motion, a closed left hand and a circle gesture with one finger of the right hand, to tell the program that the quadcopter should begin following the array's commands rather than the manual one. The commands are then extracted from the array and followed one at a time using the specific instructions of roll and pitch for each direction coupled with a predetermined flight time for which it follows that instruction to ensure that it has moved one foot in the appropriate direction. Once the array of commands in exhausted, the quadcopter will land, and is ready for continued manual flight.

Despite the fact that this all sounds very straightforward, there were a lot of issues in the physical implementation. Discovering the necessary pitch and roll percentages as well as the flight time was more of a trial and error process than a calculated one, as the small quadcopter responded to different directions in different ways. The standard four directions (forward, left, right, back) were very similar and the four diagonals were very similar, but small differences in operation made it necessary to treat each direction as its own separate entity. Furthermore, the quadcopter is rather susceptible to instability and changes in environment due to its weight and simply imperfect mechanics, so the derived commands are also inexact. This difference can be relatively harmless when travelling within one command, but when commands are carried out in a sequence and many of them are inexact, it can lead the quadcopter to stray from its path. As previously mentioned, the quadcopter has no way to sense its environment, so there is no way to check whether or not it is deviating. This leads to the quadcopter sometimes not finding its exact target location, and in some circumstances even crashing into obstacles.

7. Future Work

There are many improvements that could be made to this system. First, it would be beneficial to the Crazyflie Nano Quadcopter's ability to navigate within the physical environment if it had a sensor or

camera to help obtain information. This would allow it to interact with a dynamic environment, or simply check that it is operating correctly within the static one. Unfortunately, the quadcopter is so small and light that it limits the weight of an additions to between 5-10 g. This means that a camera or sensor that would be attached would have to be relatively simple, but still able to broadcast its input independent of the Crazyflie. This should be implemented only if a technology is ever created that has these specifications, but is also cheap enough to be worth the money and effort.

Second, a power management system would be incredibly beneficial for the Crazyflie for its thrust determination. The thrust is calculated based on a percentage of the battery power. Often during flight, the battery reports different power levels at different times due to connectivity issues, which changes the amount of thrust given and therefore the altitude of the Crazyflie from moment to moment. This makes for incredibly unstable flight, and could be improved by a thrust calculated from a formula regardless of battery level rather than directly proportional to it. This system would be less power efficient, however, and would most likely significantly lower the Crazyflie's flight time. It would therefore be beneficial to simultaneously incorporate a power failure system that is able to sense when the battery power is getting critically low. In that scenario, the Crazyflie would be able to halt commands and slowly decrease thrust to ensure that it is able to reach the ground safely rather than falling out of the air, possibly from a large height, in the case of a power failure.

8. Conclusion

Developments in motion-based technology have large implications for human-computer interaction. The technologies explored in this paper, namely the Leap Motion Controller and the Crazyflie Nano Quadcopter, have very significant computing abilities. They are quick, accurate, and agile. It is also important to note, however, that they are some of the cheaper and less critical technologies out there. They are both listed for commercial use and are available for roughly $100 apiece. Given the power that resides in such small and relatively cheap technologies, it is obvious that the power that resides in the larger and more expensive ones is therefore of great concern. As technology becomes more ubiquitous in our lives, it is important to keep track of how it is being used and to understand the implications behind its use, both in our lives and for future technological developments.

References

1. Leap Motion. (n.d.). Retrieved January 26, 2015, from https://www.leapmotion.com/product

2. The Crazyflie Nano Quadcopter | Bitcraze. (n.d.). Retrieved January 27, 2015, from http://www.bitcraze.se/crazyflie/
3. Thread: Crazyflie Nano Quadcopter. (2013, February 5). Retrieved April 30, 2015, from http://fpvlab.com/forums/showthread.php?12372-Crazyflie-Nano-Quadcopter
4. Crazyflie PC client installation. (n.d.). Retrieved January 27, 2015, from http://wiki.bitcraze.se/projects:crazyflie:pc_utils:install
5. API Overview. (n.d.). Retrieved January 26, 2015, from https://developer.leapmotion.com/documentation/python/devguide/Leap_Overview.html
6. Lester, P. (2005, July 18). A* Pathfinding for Beginners. Retrieved February 15, 2015.