Autonomous Learning in a Simulated Environment

CPR E 585X Project Proposal

Daniel Stiner, Alexander Campbell, and Chad Nelson

March 10, 2011

Table of Contents

Abstract

We propose implementation of a virtual robot intelligence; one that is driven by intrinsic curiosity and is capable of both learning properties of its environment and adapting to a variable body. The relatively simple and well-known arcade game, "Snake," will be our basis for testing the effectiveness of the project in achieving such intrinsic motivation for developmental learning. An algorithm called Intelligent Adaptive Curiosity will be used to select behaviors which result in semi-predictable, but still novel events that are optimal for learning how to better predict future changes in the environment [7]. Our hope is the continual emergence of new and more complex behavioral patterns as typically seen in previous research [5][10], despite the added complexity of a variable body schema. Furthermore, this will be accomplished on a simple and extensible framework to easily allow future research of similar motivation algorithms and other simulated environments.

Concept

I. Introduction

The Turing Test is an elementary and de facto guide for defining a computer system as intelligent. Today, 60 years after Alan Turing's first description of the test, programmers have been able to produce computing systems capable of passing restricted versions of this imitation game. One example is the Jeopardy© winning "Watson" machine which uses advanced natural language parsing and a database of facts to formulate questions corresponding with the specific answers given in the game [14]. A second example is the music composition algorithm Emily Howell who can turn out compositions by the thousands which are indistinguishable from Bach by many audiences[0] and even appreciated by professional music critics[7]. However, it is obvious that this definition of artificial intelligence is extremely behavioral-based and does not guarantee that the intelligent system be self-aware or think in a manner similar to human beings.

Unfortunately, using the Turing test for defining artificial intelligence proves too simple. Intelligence is not just producing correct behaviors in specific situations. If Jeff Hawkins' theory of the human mind being a memory-prediction framework is correct, intelligence is the general ability to recognize patterns that have been previously encountered, and relate new patterns to those previously encountered in such a way that allows for continual learning and adaptation[2]. Hawkins defines intelligence as this complex memory-prediction system of the neocortex that exists in humans and other mammals. Thus, computer systems which can accurately predict and adapt to changes in a variety of environments may be considered intelligent in different way than those which pass the Turing Test. Hawkins further argues that it will not be economical to build robots with human levels of such intelligence by current practices, rather, artificial intelligence on its current progression will be more likely to solve specific problems[2].

Oudeyer, Kaplan and Hafner expand on this in the introduction of their 2007 paper. They discuss a scenario where a computer is taught the grammar of a language by learning from a handmade sequence of increasingly difficult grammatical constructs. It develops a deeper and deeper understanding by having each successive step hand-crafted to take advantage of the prediction ability gained by the artificial intelligence system in the previous iterations[10]. This concept, developmental learning through successively more complex steps, is a key part of artificial intelligence. Yet despite this learning process being an integral part of the burgeoning field of developmental robotics, it is not without flaws.

As robots become more sophisticated and agile in the future, they will be expected to accomplish ever more complex and nuanced tasks. If the number of scenarios which must be carefully crafted continues to increase with the complexity of the actions the robots are required to perform, eventually no team of scientists or engineers will be able to teach the robots fast enough to keep pace with demand for new functionality. In addition, such knowledge and task sequences would likely be highly particular to a specific robot morphology and require specific modifications for each possible robot body to learn tasks effectively.

Looking back at Oudeyer et al. (2007), we see an alternate approach to teaching such sequentially-learning robots. Specifically, the approach has a robot use intrinsic motivation to continuously learn about the environment by seeking situations which it can almost, but not quite predict. Such situations are, in theory, the most effective way to learn how to better predict the results of behaviors and even external changes in a robot's environment. In such a model, there are two possibilities for each situation. One case occurs when the robot quickly learns to predict the outcome and thus the situation becomes very predictable and thus boring, causing it to seek a new, more novel situation. This is similar to the short attention span seen in babies for events they are capable of predicting. The other case is for events which do not show improvement in predictability over time. Such situations eventually become classified as so unpredictable as to be uninteresting, and again the desire for a more novel situation better suited for learning is heightened. This case is similar to a baby viewing "snow" on a TV. Such randomized patterns quickly loose the attention of a baby human or any other intelligent animal.

When one considers the complexity of engineering intelligent behavior using traditional approaches, the usefulness of the intrinsically motivated process becomes clear. An open question in developmental robotics today is how to create a robot capable of long-term, self-guided learning that sidesteps the problematic issue of continually engineering intelligence. Humans, throughout their lifetimes, from the neonatal stage until death, have the ability to continuously seek out and assimilate new knowledge. We feel that modeling this idea of autonomous mental development is fundamental to the success of developmental robotics. The lessons learned from our efforts today in creating a low level of self-taught intelligence have the potential to be built upon in the future to create higher levels of self-teaching robotic intelligence. These robots will be capable of not only exhibiting complex behaviors, but also making accurate predictions about the world. They will learn from self-guided experiences that will prove their high level of intelligence and understanding of the rules governing their world.

Our project aims to simulate a robot that can effectively play a computer game of Snake without any prior knowledge of the game or the rules that govern the game world. It will learn the rules from the consequences of its actions compared to its attempts to predict what will happen next for a given action. This robot will be written in C++ using the Allegro graphics library. The code of the robot's "mind" will be abstracted into several modular boxes. The first box will embody the robot. It will attempt to predict future states of the game world given a history of the current and previous states. The first box will also be responsible for selecting an action for the robot to perform. The second box will actually control movement and other behaviors, using current data from the environment and motor commands from the first box. Various algorithms may be used for this controller and the predictor modules. The predictor will likely be based on artificial neural networks, but the controller may be anything from randomized movement babbling to a pre-programmed rule based system. Our main interest however is a control algorithm which consistently seeks novel inputs for the prediction system to analyze. In theory selecting such actions at first will be impossible and the system will degrade to simple movement babbling. Yet if the prediction module works correctly for our domain, the snake will exhibit forms of coordinated movement.

In summary, the goal of this project is to model the learning abilities of a two year old child[10] playing the simple computer game of Snake, given that the child has direct input through a joystick or other simple means to control movement of the snake. This will provide a viable platform to compare different methods of motivation and learning. However, we will only examine the behavior of a select few promising curiosity algorithms in this project due to time constraints.

II. Prior Work

The concept of intrinsic motivation resulting from a desire to explore the world is not a recent thought. Robert White argued in essays as far back as 1959 that an internal drive exists beyond the traditional primary biological drives such as hunger and thirst. White called this drive "Competence Motivation," the idea that animals are driven by a constant internal need to interact more effectively with their environment[11]. Combining this need to continually learn how to more effectively manipulate the environment along with the previous definition by Hawkings[2], creates a theory that developing true intelligence is possible by a system that continually desires at least in part to explore the world and find situations from which it can learn to better predict the consequences of its actions. Such a robot fulfills the ideals of both White's and Hawking's theories.

Despite the relative simplicity of this concept, only in the last 20 years have scientists such as Oudeyer[10], Marshall[6], and Schidhuber[12] gone beyond thought experiments and started to develop descriptive algorithms, working simulations, and tangible robots. What follows is a listing of many different strategies taken by robotics scientists to this concept of curiosity-based motivation in creating intelligent robots. The goal of this project is to further current research by looking at variations of approaches that are thought to be feasible, yet do not have a large number of papers detailing them and the behaviors they can produce currently.

A paper on Intelligent Adaptive Curiosity defines IAC as "a drive which pushes the robot towards situations in which it maximizes its learning process. It makes the robot focus on situations which are neither too predictable nor too unpredictable.” [8] This idea will undoubtedly play a crucial role in our project.

The following algorithms (explained by Oudeyer and Kaplan[9]) may be seen as different approaches to how the robot may focus on desirable situations and each will be covered in greater detail in the algorithms section. The authors also compared many of the currently existing methodologies for intrinsic motivation, many of which are driven by curiosity. These ideas support the view that using this kind of exploration in robotics is both promising enough to be worth investigating and has room for novel research. From Oudeyer and Kaplan’s descriptions and other rankings we have drawn three of the overall best and least researched curiosity algorithms.

The simplest algorithm is Information Gain Motivation, which rewards a robot for learning. A typical implementation will check the accuracy of robot’s predictions against the actual outcome of events, repeating behaviors which are more predictable than a specified lower bound but yet less predictable than a certain upper bound. The simplest way to do this is to take the ratio of successful predictions over total predictions to create a percentage that is easily bounded.

A slightly more complex algorithm called Learning Progress Motivation favors learning to complete a task more efficiently by either solving a problem faster. One problem however, is the definition of efficiency which requires quantifying a measure of progress.

Finally, the concept of Flow Motivation, more formally called Competence Progress Maximizing which is modeled after the concept from psychology of being “in the flow.”

III. Qualifications

Alex has done quite of bit of computer 2D animated graphics using C++. He’s been programming for five or six years in many different programming languages (Python, Basic, Pascal, and of course C/C++). He took Professor Stoytchev’s Spring 2010 course on computational perception, experience which just might come in handy during this project. Alex’s interest in programming stems from his desire to develop the next generation of video games.

Daniel started programming Flash animations and simple web pages in the 7th grade and has since developed a bit of experience and a love of writing code. With recent classes in C++ and Java and a further smattering of languages used on small personal projects, he feels up to the programming required to carry out this project. Furthermore, he has previously read into the workings and applications of artificial neural networks in the context of artificial intelligence. Such networks will likely be used in some of the prediction algorithms of this self-guided learning approach.

Chad’s skill set involves multiple years of experience programming. He has created a 3D version of snake in the past using C++ and GLUT. Additionally, he has a great familiarity programming in the following languages: Java, C#, C, C++, Objective-C, Python, PHP, and ActionScript 3.

Approach

IV. Game Specifications

We are defining a simplified version of the game Snake to use as a standard test for our framework. It will be very similar to the game traditionally played in arcades around the world and popularized by mobile phones[4], but with modifications to make algorithmic prediction of the game environment easier and accidental death by the snake more difficult. Traditionally, the game has these properties[15]:

- To start, the snake is a single block in the middle of the screen
- The snake can move in the four cardinal directions
- The game consists of a n * m grid with each square being snake, apple, or blank
- There is always one apple on the grid, and only one apple
- At each time step the snake “slides” in the selected or last chosen direction
- “Eating” an apple increases the length of the snake by one
- Moving the head of the snake into the body results in death/loss
- If any part of the snake moves off the board, death/loss also occurs
- If you lose a life the game is over. 25 ¢ to play again!
- Points are awarded for eating apples

The main modification we propose to make the game easier for the robot is to remove the boundaries at the edges of the board. As the snake leaves one side of the playing field, it will emerge from the opposing side heading in the same direction. This will reduce the number of cases where death is possible, even to the point of death being impossible with a one length snake consisting of a single block. This gives the robot a chance to explore the affordances of the game world without being reset continually by hitting an edge. However, as playing progresses to a snake of length five, the death affordance finally reappears with the ability to contort the snake in such a way as to hit its own tail and die. Thus this modification does not change the game in the long term, but gives the robot time to learn in a safe environment and start to exhibit observable learning behavior.

Other modifications proposed are not essential to the success of our project, but are for comparison’s sake to support the generality of our approach to learning. In all cases these modifications are interchangeable, and if one is greatly beneficial to learning performance we may include it in a final hybrid environment which maximizes learning potential. This will be the version that we showcase. The first modification is related to the movement behaviors of the snake. Instead of the traditional selection of a cardinal direction to move, direction can be more simply controlled by selecting to turn left or right, or to continue in the same direction. Second is to not use a discrete grid, but to have a continuous playing field more analogous to real life where a grayscale image of the game field is captured and fed into the robot’s mind. Lastly, is the extension to a more complex environment containing multiple apples, or other kinds of fruit which will have various effects on the snake.

We want the artificial intelligence to require as little pre-programming as possible. In order to achieve this goal, the robot will be unaware of the meaning attached to the different inputs that are given to it, knowing only the raw data. A major question, given our project, is will the robot will be able to assemble the inputs into the proper configuration without pre-training?

V. Algorithms

The paper “What is intrinsic motivation? A typology of computational approaches” [8] provides an excellent overview of the various methodologies for implementing intrinsically motivated learning. The paper also supports our view that these kinds of exploration are both promising enough to be worth investigating and new enough to have room for novel research. Near the end of the paper, the authors rank each of the examined motivation algorithms according to various criteria. From this we have drawn the three overall best and least explored methodologies.

The first algorithm is called Information Gain Motivation. This algorithm attempts to simulate the basic pleasure of learning. Information Gain Motivation is useful in a few niche cases, but it is usually better if combined with another algorithm. One reason Information Gain Motivation can be useful is that it can be measured simply by checking the robot’s predictions against real events. This will give solid evidence that the robot is learning. For more information on Information Gain Motivation, see Fedorov’s 1972 paper on the subject[13] as well as the paper by Roy and McCallum from 2001[1].

The second algorithm, Learning Progress Motivation, rewards the robot for doing a task more efficiently, either by solving a problem faster or learning to perform a task. The main road block in LPM is the difficulty in quantifying “progress”. We can simplify this problem by only looking at a single context, vision. Two papers dealing with this algorithm are the paper by Oudeyer et al. (*Mean error for last tau contexts,* 2007) and the paper by Schmidhuber, (*Simply compare prediction of current state before and after a training*, 1991).

The third and final algorithm is Competence Progress Maximizing - also known as Flow Motivation. As an example, imagine you are working on a problem and are going along swiftly without any hitches; you are said to be “In the flow”. When you are “In the flow” a single distraction can completely destroy your concentration and stop the tremendous burst of speed. Afterwards, it is difficult to re-achieve this state. As a possible extension to CPM we could simply encourage the robot by larger and larger amounts the longer it keeps making the right decisions. When it fails the streak is broken and it is no longer encouraged. Rewarding the snake in the way makes it favor sequences of optimal decisions.

Libraries

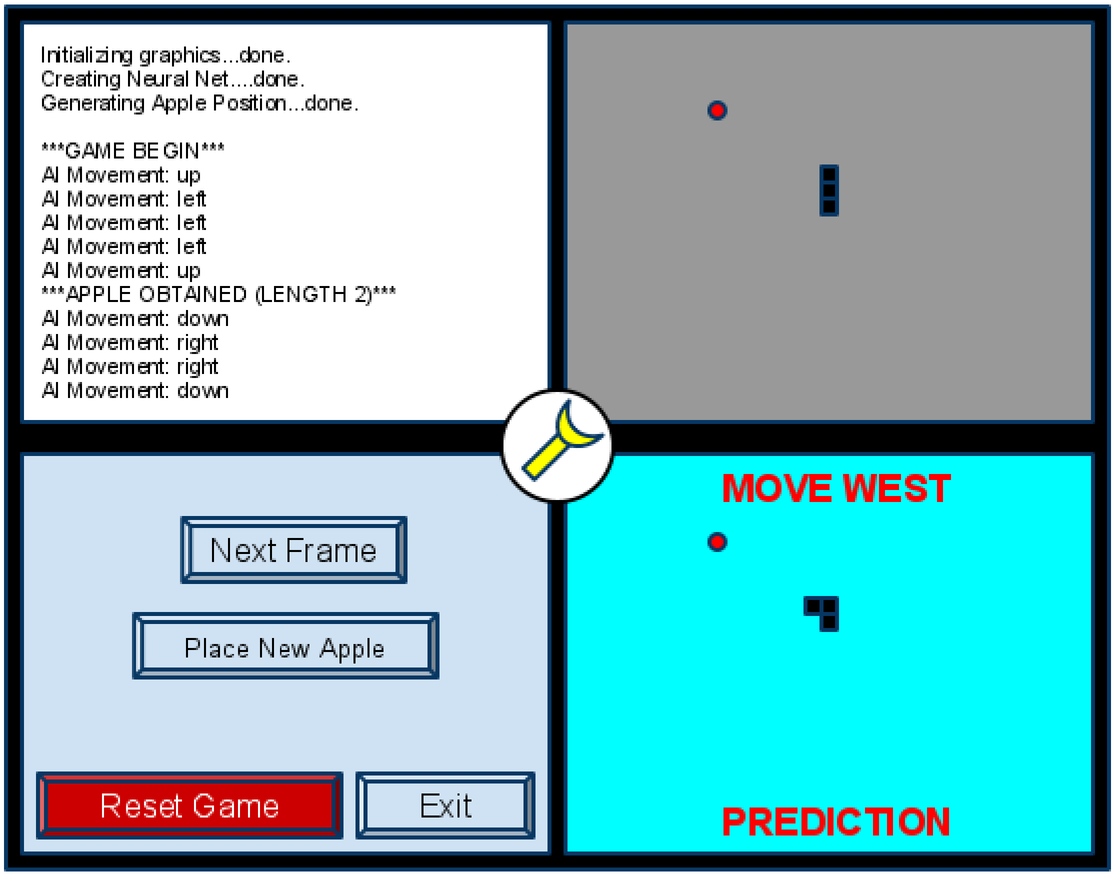
For our project, we will combine the power and object-oriented capabilities of C++ with the simple and fast graphics/sound/input library of Allegro. Because both C++ and Allegro are cross platform and aren't taxing on the system, our application will work well on most modern computers. We'll also be utilizing Allegro's sound capabilities, adding a second modality that the robot can use as input. It is our hope that the robot's perception will increase by factoring sound into the gameplay (beeps and blips for eating an apple and other interactions).

VI. User Interface

To maximize the amount of information shown at one time on-screen, we will divide the window of our application into four quadrants. These quadrants will be:

- The current state of the game itself.
- Console output.
- The current prediction of the next frame of the game along with the direction the robot has decided to move.
- A user interaction window where it is possible to reset, quit, and halt the game. Also in this quadrant is an option to manually add fruit to the game.

See the mock-up below for an example of a possible setup:

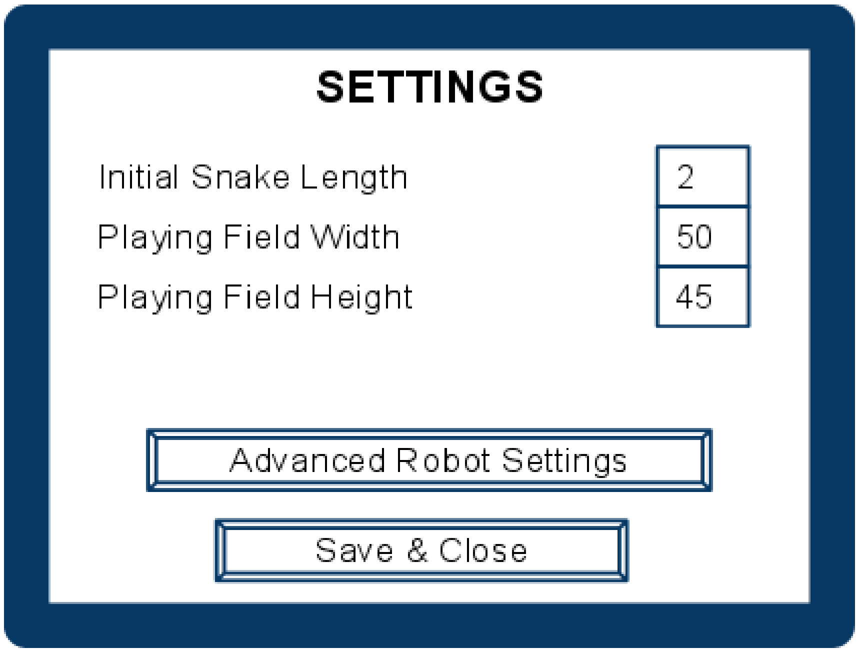


```
Initializing graphics...done.
Creating Neural Net....done.
Generating Apple Position...done.

***GAME BEGIN***
AI Movement: up
AI Movement: left
AI Movement: left
AI Movement: left
AI Movement: up
***APPLE OBTAINED (LENGTH 2)***
AI Movement: down
AI Movement: right
AI Movement: right
AI Movement: down
```

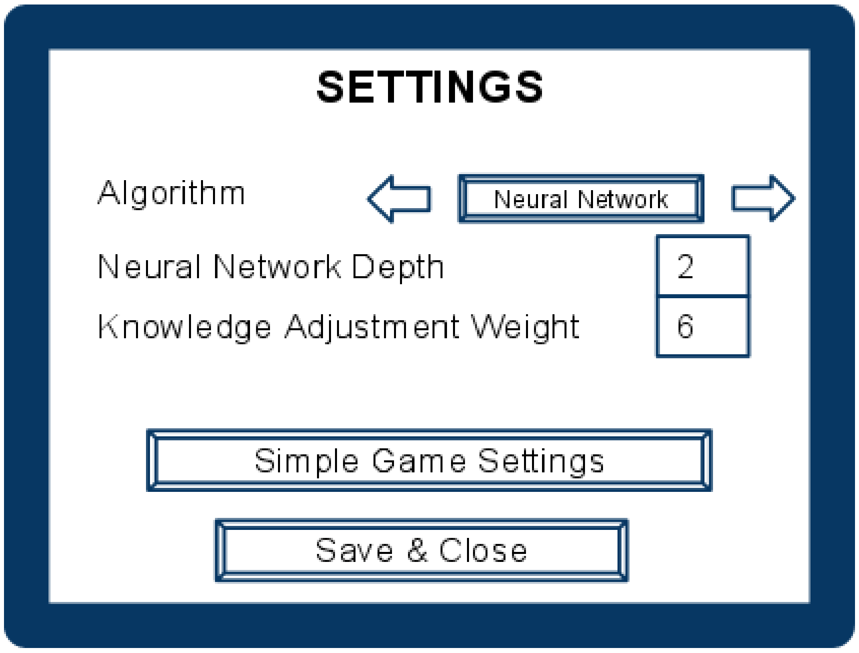
Because repeatedly clicking "Next Frame" may wear out your index finger (or, if you are missing your index finger, your middle finger) holding the space bar will advance the frame at a rate of five times per second. Pushing the escape key at any time during the game will bring up a dialog box that asks whether the user wishes to exit immediately or not. Pushing "R" will immediately reset the game without asking for confirmation. All keystrokes and other design elements are subject to change during the programming process.

In the center of the screen will be an icon of a wrench signifying settings. Clicking this will open a settings box, somewhat like the following:


SETTINGS

Initial Snake Length 2
Playing Field Width 50
Playing Field Height 45

The "Advanced Robot Settings" button would toggle between the current settings box and the following box, which allows for fine-tuning of the AI algorithm:


SETTINGS

Algorithm Neural Network
Neural Network Depth 2
Knowledge Adjustment Weight 6

The algorithm section will allow you to switch between the various neural network-style algorithms and a traditional Snake-playing bot. If time permits, a third algorithm will run the real robot using a Barret Arm and a standard joystick with the same AI algorithm in use with the virtual one. The game would be projected onto a screen in front of the robot, while the UI described above would be run on a separate monitor with human surveillance (see the mockup below).

This algorithm may have trouble with glare and other noise so success rates are unlikely to be as high as with the virtual robot, but it would still be a very interesting extension of the experiment in the physical world.

Robot concept art by Izaak Moody [3]

VII. Implementation Timeline

The following is a timeline for the proposed project. It is subject to change, and we may have to adjust the timeline if it turns out one section takes more time than initially expected.

Note: A ✓ beside an item means it is already completed.

✓ Give the initial presentation.
✓ Write this proposal
 Work on the project
- Week 1 (Spring Break): Read additional related papers to gain additional depth of understanding of previous work in the area and confirm validity of proposal and methodology
- Week 2: Working snake simulation playable by human and rule based AI
- Week 3: Prediction neural network playing the simulation. Start paper.
- Week 4: First draft of paper
- Week 5: Finish project paper
- Final Week: Buffer Week

 Present the project in class

Evaluation

VIII. Goal

Mountcastle's 1957 paper about the columnar nature of the cerebral cortex revolutionized science's understanding of the brain; that a standard, repetitive structure exists throughout the neocortex. Not only does a similar structure exist throughout the brain, but, according to Hawkins, the brain also has a similar algorithm for learning[2]. Experiments using ferret brains and other animals have shown that if you reroute the visual senses from one part of the brain to another, the visual center of the brain will move to the new location. This evidence, as Hawkins suggests, seems to support the single algorithm theory of the brain. The implication for robotics is that intelligent behavior can be created by mimicking the way brains work using computers.

The goal of this project is to create a simplified version of the algorithm that exists in the brain, and use it to create a low level of intelligent behavior. In the case of this project, intelligent behavior will be defined as behavior that uses past experience to make predictions, and the use of these predictions to achieve a high game score.

Success in our case is partially defined as creating a robot that will repeatedly chase the apples, and grow to a length such as ten. The other part is comparing it to a control group as outlined in the next section. If it turns out that this goal is easily reached, we may attempt having a physical robot play the game. This robot would use a camera to watch the game being played on-screen, and run a joystick with its hand.

IX. Evaluation Methodology

Two controls will be used to quantify the effectiveness of our artificial intelligence: random movement babbling and a simple rule-based AI. The difference between these control runs and the AI robot will be quantified by several simple statistics such as the score achieved in an average round of play, the lifespan of the snake, and the average number of steps taken to find each successive apple.

A second set of measurements for observing the rate of learning is also proposed. We will compare time spent at each length of snake to see if prediction algorithms' time to learn a new body schema depends on its complexity. The complexity of course will increase with the length of the snake. Looking specifically at the rate at which survival time increases each time the snake dies, and after it has completed another learning phase (i.e. 10,000 steps of simulation) should see a logarithmic or similar curve that levels off as the difficulty of gaining more length increases.

Specifics of how to compute averages for these measures have not been decided. Based on research involving artificial neural networks, running one thousand simulations for each control algorithm on a timescale the order of ten-thousand steps and taking the mean of the results seems reasonable and should not take an exorbitant amount of computational time given the limited number of behaviors, relatively small size of the game world, and simple algorithms written in low level code. Such a large number of results should give us a high chance of finding statistically significant differences in performance measures using a test such as the p-test for the different algorithmic approaches.

X. Conclusion

By modeling the control of a virtual robot after the intelligent mechanisms that already exist in biology, this project will push the frontier of developmental robotics further by demonstrating a viable solution to the problem of goal oriented behavior motivated from intrinsic curiosity. Also, the clean interface and logical design will allow even novice users to watch as the power of prediction based artificial intelligence is put to the test in a game they know and have played.

If we do attempt to make a physical robot play Snake (see the part about this in the UI section) it would be the first on-the-fly learning of a video game by a physical robot, and therefore, truly a novel experiment.

XI. Future Expansion

The easiest future expansion of the project would be to branch out and have our algorithm learn to play more games, such as Pong, Asteroids, Tetris, or Pac-Man. It might also be possible to use the same approach to other, bigger, more complex games like *Wolfenstein 3D* (© iD software). However, moving to a three dimensional world might require a significant leap forward in the visual processing capabilities of the robot.

Other future research includes attempting to play video games by having a physical robot use a joystick. Our project could be split into the simulation running on a TV screen with the control algorithm running on the robot. The controller would initiate arm movements of the robot directly, based on unfiltered inputs of visual, sound and proprioception data. Thus the robot could attempt to play simple games in a controlled environment. Furthermore, success at playing such a simple game could allow for further expansions into exploration of individual objects, where easily recognizable or predictable objects are examined first, but quickly become uninteresting and cause the robot to efficiently explore through simple play the set of objects given to it.

References

[0] Blitstein, Ryan. "Triumph of the Cyborg Composer." *Smart Journalism. Real Solutions. Miller-McCune.* 22 Feb. 2011.
http://www.miller-mccune.com/culture-society/triumph-of-the-cyborg-composer-8507

[1] Fedorov, Theory of Optimal Experiment. New York, NY: Academic, 1972.

[2] Hawkins, Jeff, and Sandra Blakeslee. *On Intelligence*. New York: Times, 2004. Print.

[3] Isaak Moody. “Robot Concept Art Images.” http://home.engineering.iastate.edu/~alexs/lab/media/images/concept_images/

[4] James. "History of Nokia Part 2: Snake." Nokia Conversations. Nokia, 20 Jan. 2009 http://conversations.nokia.com/2009/01/20/history-of-nokia-part-2-snake/

[5] Marshall, Blank, and Meeden; An Emergent Framework for Self-Motivation in Developmental Robotics
http://www.cs.swarthmore.edu/~meeden/papers/marshall.icdl04.pdf

[6] Marshall, Blank, Kumar, and Meeden; Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture
http://www.cs.swarthmore.edu/~meeden/papers/blank.cybersys.pdf

[7] Morrison, Richard. "Emily Howell, the Computer-composer, Cannot Match Human Inspiration - Times Online." *The Times | UK News, World News and Opinion*. 22 Oct. 2009
http://technology.timesonline.co.uk/tol/news/tech_and_web/article6884606.ece

[8] Oudeyer, P.-Y. and Kaplan, F. (2004). Intelligent adaptive
curiosity: a source of self-development. In Proceedings of the 4th International Work-
shop on Epigenetic Robotics, volume 117, pages 127–130.

[9] Oudeyer, P.-Y. and F. Kaplan. "What is intrinsic motivation? A typology of computational approaches", Frontiers Neurorobot., pp. 2007.

[10] Oudeyer, P.-Y., F. Kaplan, and V. Hafner (2007). Intrinsic motivation systems for autonomous mental development. IEEE Transactions on Evolutionary Computation, 11(1):265–286.

[11] Robert W. White, Motivation reconsidered: The concept of competence, Psychological Review, 66:297–333, ISSN 0033-295X, DOI: 10.1037/h0040934.

[12] Schmidhuber, Jürgen. "Developmental Robotics, Optimal Artificial Curiosity, Creativity, Music, and the Fine Arts." Connection Science 18 (2006): 173-87.

[13] Roy and A. McCallum, “Towards optimal active learning through sampling estimation of error reduction,” in Proc. 18th Int. Conf. Mach. Learn., 2001, pp. 441–448.

[14] "Watson Algorithms Team." *IBM - United States*. http://www.ibm.com/innovation/us/watson/research-team/algorithms.html

[15] "Blockade Video Game, Gremlin Ind, Inc. (1976)." Arcade-history.com. 4 Apr. 2008 http://www.arcade-history.com/?n=blockade&page=detail&id=287