# Donald Nute's VWorld

## *Bumble 2.0*

---

*Author:*
**James P. Spencer**
Computer Science (Games)
Student Number: 08809130

*Author:*
**Thomas J. Taylor**
Computer Science (Games)
Student Number: 08813043

---

School of Computing, Engineering and Mathematics
**University of Brighton**

Documentation

April, 2012

# Introduction

At the beginning of the module, we were presented with Donald Knute's VWorld: a simple 2D sandbox game environment written in the Prolog logic programming language.

The game world itself is represented as a 2D grid structure, and can consist of a number of different levels (or rooms) interconnected with doors. Each level is filled with a number of different objects, ranging from impassable walls, to collectibles such as health power ups (fruit) and weapons (a sword), to NPCs such as snails and hornets, which try to hinder the player's progress by inflicting damage or blocking their path.

The VWorld game framework we were given was already set up to deal with the more low-level functionality such as the user-interface elements and graphical display and loading and initialising the game. A number of different behaviours were also predefined for the non-player characters (NPCs). For example, the NPCs were able to move around the level (and guard objects in the case of Dragons and witches). As mentioned, the snails follow the player in an attempt to block their path, and the hornets attack the player, inflicting damage.

The main protagonist of the game is Bumble: an AI character who finds himself in the mysterious realm of VWorld. He must have his wits about him at all times in order to survive this treacherous land. Bumble is given a number of tools to aid him in his fight against evil, such as a sword, a shield and bugspray. These items are scattered about the level, and must be searched down before they can be used.

# Aims and Objectives

We were given a lot of freedom with this project in terms of the actual requirements. As there is no real 'aim' in the VWorld game, we were left to our own devices to develop the system as we saw fit. The guidelines for the project were appropriately open-ended:

> Modify aspects of [*VWorld*] to make the behaviour of the actors (including Bumble) more 'interesting' than it is at the end of the formal workshops

To give us some idea as to which areas we could develop, we were given a number of test level maps, which are designed in such a way as to encourage certain behaviour. For example, some of the test levels included the 'bird' character who must be caught to complete the level. Catching the bird requires Bumble to collect 'birdseed' located somewhere in the level, and so it can easily be seen that some sort of objectives are required for Bumble to succeed in such levels. Similarly, some levels have locked doors which require specific keys to enter.

There were a number of different angles from which we could approach this problem. Our first goal with this project was to simply increase Bumble's survival rate, as we felt that this was needed before any other aspects of the game could be improved. As a part of this objective, we also wanted to fix a number of issues with the basic Bumble and VWorld framework. Moving on from these small incremental changes, we wanted to implement some sort of search algorithm which Bumble could use in combination with his dynamic map of the world to find objects he had previously seen.

We used the test levels to create a basic 'specification' of the kind of behaviour we wanted Bumble to exhibit:

- Some form of search
- Refine Bumble's strength/damage system
- Implement some exploration

Our overall goal was to iterate rather than to innovate, by developing the existing behaviour, fixing existing issues, and by implementing more incremental changes.

## Basic Bumble

To start us off, we were given a very basic Bumble which we could use as a starting point for our development. The original Bumble was able to randomly explore the environment, but little else. In the tutorial sessions, we began to develop some basic behaviour for Bumble. By the end of the tutorials, Bumble was able to:

- Randomly navigate the world
- Pick up any interesting/desirable objects he was stood next to

By the end of the tutorials, Bumble was able to move around the game world, and build a 'map' of where certain objects were located as they were found.

Bumble has a few basic stats which are updated as he roams the level: strength, which decreases as Bumble moves around the level and can be replenished with tasty apples, and damage, which is incremented if Bumble is attacked. Bumble dies if either his strength reaches 0, which means he is unable to move, or if his damage reaches 100%.

Bumble also had a few basic rules to help with his survival in the dangerous world. These rules are used to ascertain whether Bumble is hungry or hurt based on his current strength and damage levels. This information is then used to point Bumble in the direction of any power-ups as appropriate. This behaviour was quite naive, as Bumble would only move towards power-ups in his immediate vicinity (i.e. less than 2 squares away). In addition to this, Bumble also collected any 'interesting' or 'desirable' objects that he was standing next to.

As it was, Bumble's behaviour was still very naive. Bumble only considered objects which were very close by (i.e. within a two square 'radius'). Although we had a map of the world which Bumble created dynamically as he explored the world, nothing was done with this map. The way Bumble treated objects in the game was also very simplistic: 'interesting' or 'desirable' objects were only picked up if Bumble was stood directly beside them, and Bumble had no real decision-making process with regards to picking up fruit or crosses. As it stood, Bumble would simple become 'hungry' when his strength dropped to a certain level, and 'hurt' when his damage exceeded a set value; there was no reasoning behind his decisions. Additionally,

; he was only able to react to objects he was close to, despite the fact that a map was dynamically generated as Bumble explored the level. There was also no real 'method' behind

Bumble's navigation, he simple wandered around randomly. Bumble also had no sense of any 'objectives' in the level. For example, he had no idea that in order to catch the bird, he first needed some birdseed. There were also a number of small bugs in the existing system which hindered his progress.

## Modifications

One of the key elements in our strategy was the smaller, more incremental changes and bug-fixes to the original system. While these may seem fairly insignificant, we feel that they actually contributed greatly to Bumble's appearance of intelligence, and the overall 'believability' of VWorld as a game.

One of the first changes we made was to add extra predicates to allow Bumble to spot interesting and desirable objects from two squares away, and move towards them (as he does with the tree/cross). This obviously meant that Bumble was more likely to pick up these useful objects whilst exploring. We also ordered these predicates in a logical was to reflect the priority that Bumble would use. The priority for removable objects (including crosses) below:

•

health and strength power-ups are considered to be the highest priority. However, we consider any 'nearby' objects to be of a higher priority to those Bumble

The hornets follow Bumble upon spying him.

Bumble no longer flees if in possession of the shield/sword

When Bumble becomes hurt, he checks the map for the appropriate resources to fix him. Fixed a bug whereby Bumble could get stuck alternating between two interesting objects (e.g. tree/cross)

Hornets no longer cause injury if Bumble's in possession of the bugspray. Also, the bugspray now actually works.

Added a conversion from cartesian coordinates to directions

Bumble now spies desirable/interesting objects from 2 squares away

Fixed a bug whereby Bumble could get stuck in a corridor 1 sq wide (see castle2.vw). If there was an interesting/ desirable object at the end of the corridor, Bumble would try to go west, but would simple stay in the same sqaure. There's now a check to see if Bumble already attempted the same move, if so, he moves randomly.

A map of the entire level is built. When Bumble goes through a door, an extra square is added in the appropriate direction to account for the wall in-between the levels.

As removable items are taken by Bumble, they are also removed from the map.

Implemented basic fuzzy reasoning for the strength/health. Bumble now has more levels: Not hungry, small hungry, medium hungry, starving etc. These levels are used when determining whether Bumble should consume fruit/use crosses. If Bumble sees fruit, but is very hurt, it will pass by and continue looking for a cross.

Added an exploration mode which is enabled/disabled based on Bumble's current health/strength. If not hungry/hurt, Bumble will explore the map for areas not visited using the preference:

e, nw, se, sw, n, e, s, w. As Bumble explores, the squares he visits are marked off, removing them from the exploration mode. Bumble stops exploring if he can't see any nearby unvisited squares.

# New Features

## Search

Search is a major area of game AI as it helps add realism and effeciency to agents movements.

In VWorld if bumble is to survice an negate the world effectivly it is vital for him to have some form of search so he can navigate back to an object he has seen for instance if he is hungry he should negate to a tree or any food he has seen.

We have experimented with implemeing several types of search, this has been done through carrying out the activities outlined in workshops 4 and 5.

Our inital investigations explored the use of breadth and depth first search, both of these types of search arn't well suited for our gamewold and games in general, the main resoning for this is their extreamly high worst case perfomance and the fact that they don't produce natural or effectient search paths.

It is for this reason we explored the use of heuristic based search, where we looked at greedy and A* search although it should be noted we did consider several other search methods that could give naturalistic approaches such as Theta* however it was demed implementing this would be nontrivial and of little value due to the grid based system present in vworld.

Overall Implemening search has proven to be challenging and for reason we have been unable to implement A* instead sticking with greedy search. At the highest level our search is functioning by looking for a parth to an object given a current condition, calling the search predicate will generate a path to the desired object if it is contained in bubmles map of the current world. There are however issues that can arise in this proccess, the first issue arises when another agent such as a hornet blocks bumble's next step, in this situation he will have to avoid the desired cell and move into one adjacent to it. This has been handled by defining a list of predicates for adjacent cells one such example of this is given our desired cell north west and it is blocked by a hornet bumble will move to either north or west if he can, afterwhich he will ease his current path. If bumble is unable to follow the specified path and move to an adjavent cell he will erase his current path. Essentially this is a simple form of collision avoidance this could be extended furthur through the use of ray casting to achive an effect similar to that of open steer.

Our implementation of greedy search is based off the code provided for the workshop we have had several issues with this code and for this reason we have made some minor adjustments. The main change we have implented is a counter on the amount of times the search can recuse, this is considered a hack however we have deemed this to be nessacery to prevent the heap from overflowing and crashing the application. We had tierlessly attempted to find and debug the underlying cause of this issue. Although this bug does not fully prevent search from functioning, it will however mean that the application is unlikely to run for a great deal of time as bubmle will often become stuck and die of starvation. Overall this

bug is primary barrier to hightend success of the system and in resolving it we feel that we would have had a far more impressive submission.

### Fuzzy-Inspired Reasoning

Didn't bother with the probability component, as it caused unpredictable and unrealistic behaviour.

## Team Development

## Testing and Results

We have utilised the set of test maps supplied with bumble to evaluate the effectiveness of our solution. In this experimentation we work with the assumption that if bumble survives more than 5000 moves he will live infinitely.

???3 or 5 tests ?

| Map | Test 1 | Test 2 | Test 3 | Mean |
|---|---|---|---|---|
| Test 1 | X | Y | Z | X+Y+Z/3 |

Table 1: This table shows some data

## Analysis

What we found out from the results

## Conclusions

## Further Work

### Machine Learning

We have both completed final year projects in this area, and this is something of great interest, we feel that this is very interesting area of AI...

#### Q-learning

#### Reinforcement Learning

#### Genetic Algorithms

## Search Improvements

At current the search in the system is simple in nature, it was desired to have implemented A* but this has unfortunately proven to be problematic.

The current search could be improved by searching for the same goal once a search is erased .

Collision avoidance.

Naturalistic search.

## Mapping Improvements

The mapping algorithm is simple in nature and could be improved by the use of?

Boundary Detection (IE) can i see a wall ? is the cell adjacent a wall ? therefore i do not need to visit that cell because it won't reveal anymore information to me.

The problem of mapping is strongly linked to Maze exploration, there are several interesting maze exploration algorithms such as (Azkaban algorithm, Dead-end filling, Wall follower ) and we feel that the use of one of these would make the Bumble's discovery of areas more effectively.

Out of these of we feel that the wall follower algorithm is well suited to bumbles world would....

Talk about wall follower.