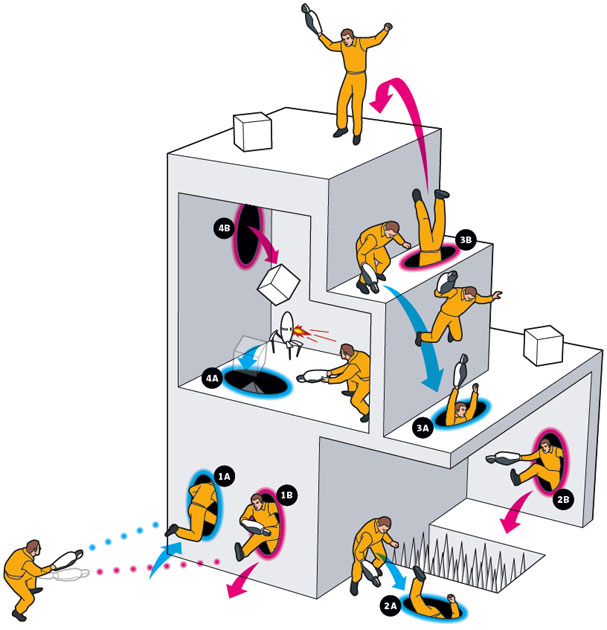
**BLASTRS**

**Dev Diary- Nikhil Suresh**

**Some inspiration**

This game idea is inspired by Portal, an action puzzle game created by Valve.



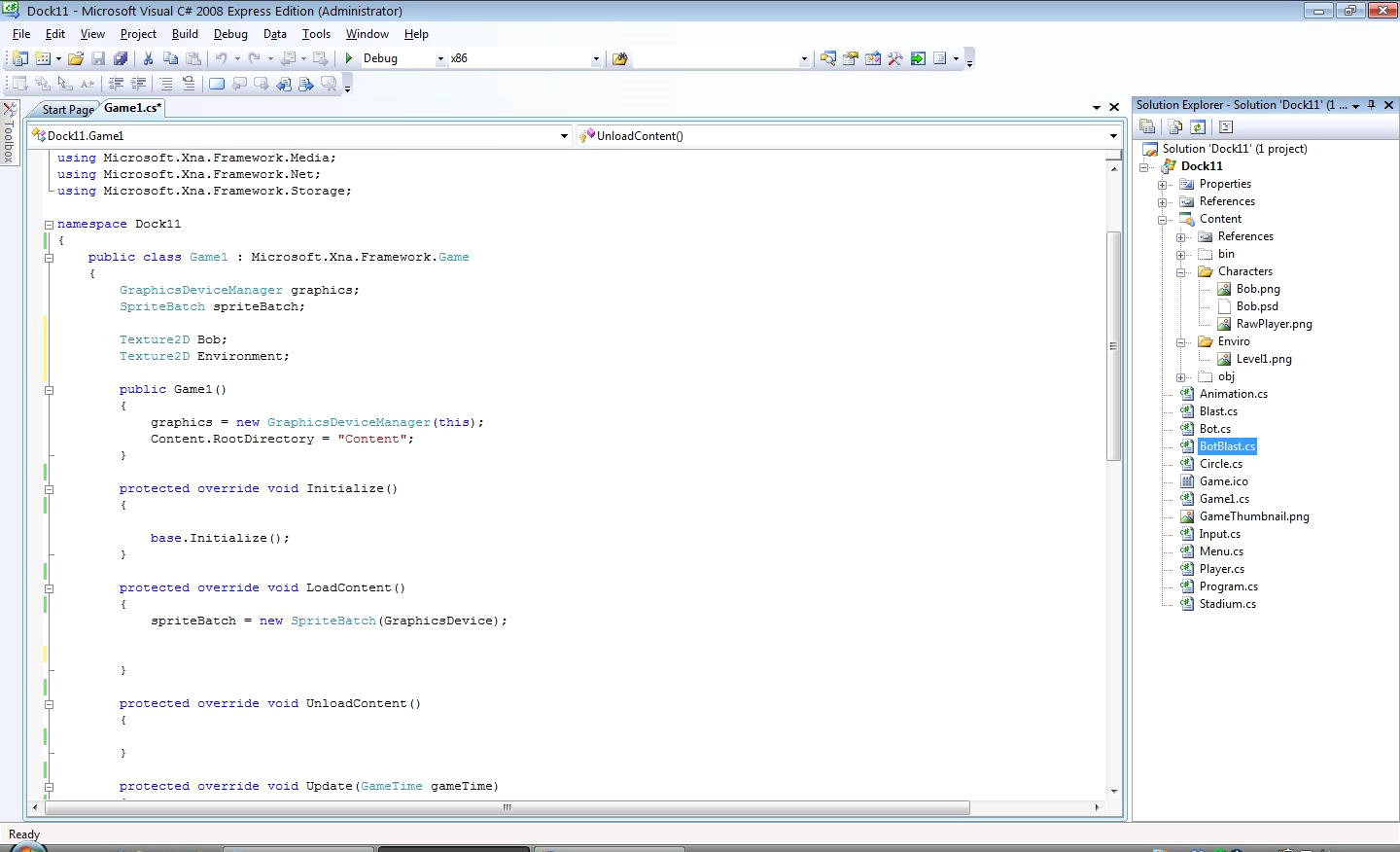
This game will use a game play mechanic called Blasting, where the player is able to manipulate objects and push them away.

**(Not) Starting from scratch**

Every project starts from somewhere. This one is, fortunately, going to be using a lot tech I've worked on in the past, so I can remember code, or if required, even inherit classes for the following objects:

* Players
* Levels
* Menus
* Blasts

The Blast mechanism was used in a past production, 'blastrs', which was entered into a game development competition. I fully developed the required classes myself, so copyright issues should not be a problem.

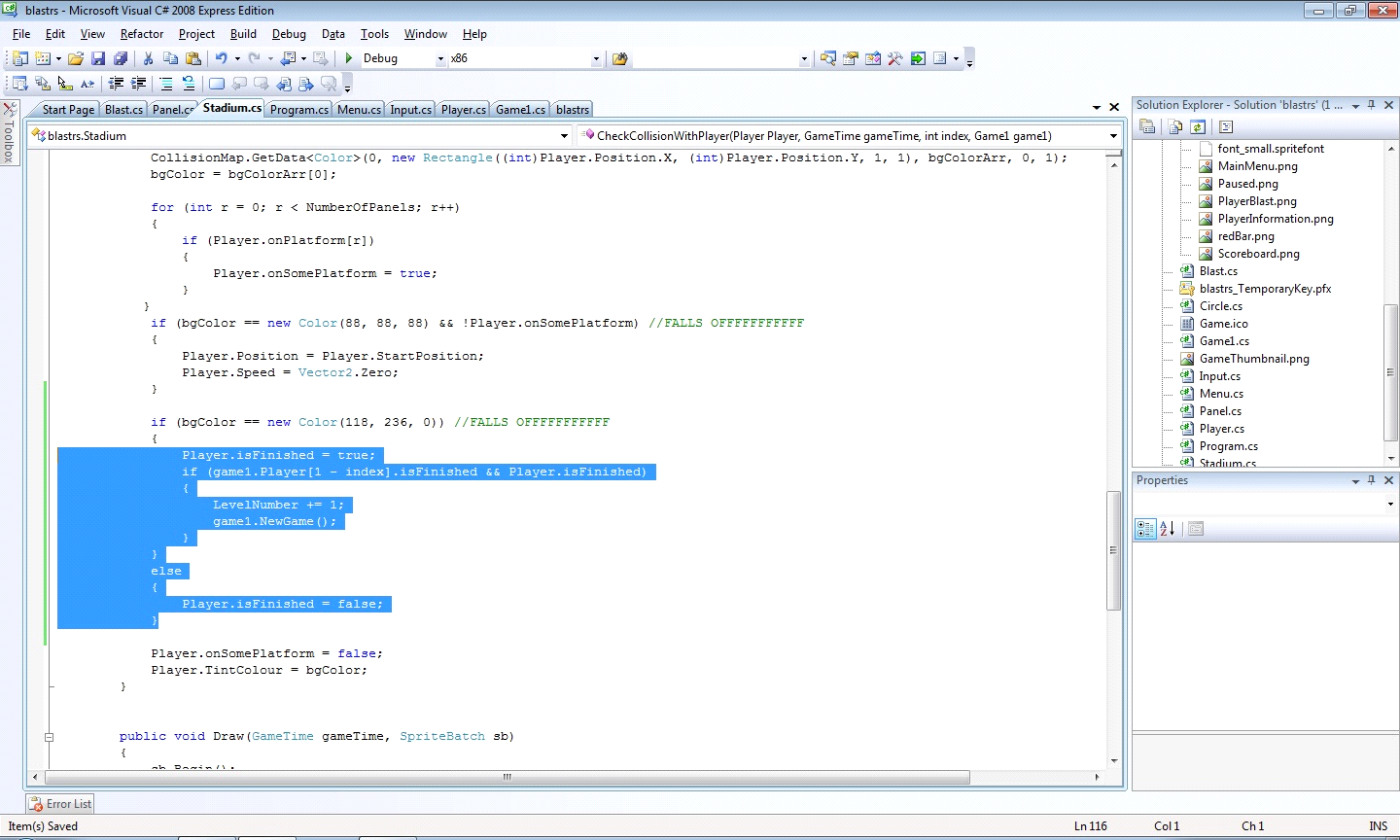


**Which Player Do I Choose?**

I really want to make at least some part of the game a single-player experience, but I could not resist the temptation of designing a couple of multiplayer based levels.

Therefore, I've had to implement 2 players in the game. This was not a hard task, seeing as it is simple to instantiate a new player. However, this means that the game logic must work with the second player as well.

One such problem I faced was a player selection requirement. For each blast, I had to identify which player was not being affected. I devised an algorithm to find the index of the unselected player.



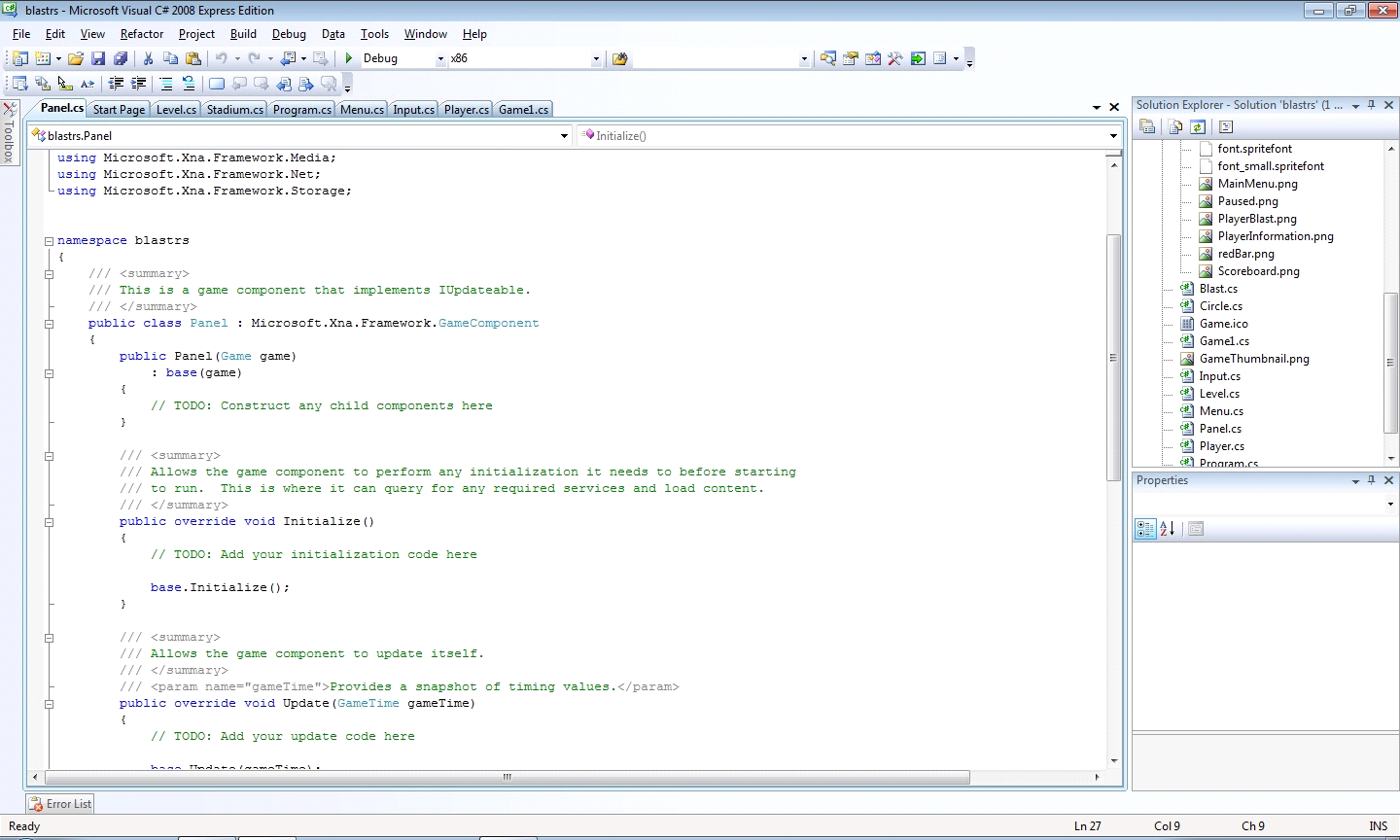
If there are 2 players, Player 0 and Player 1, subtract the player number from 1:

* If **0** is selected, the other player is 1 - **0** = 1; Player 1
* If **1** is selected, the other player is 1 - **1** = 0; Player 0

Since the game will only have a max of 2 players, this algorithm doesn't need to be expanded.

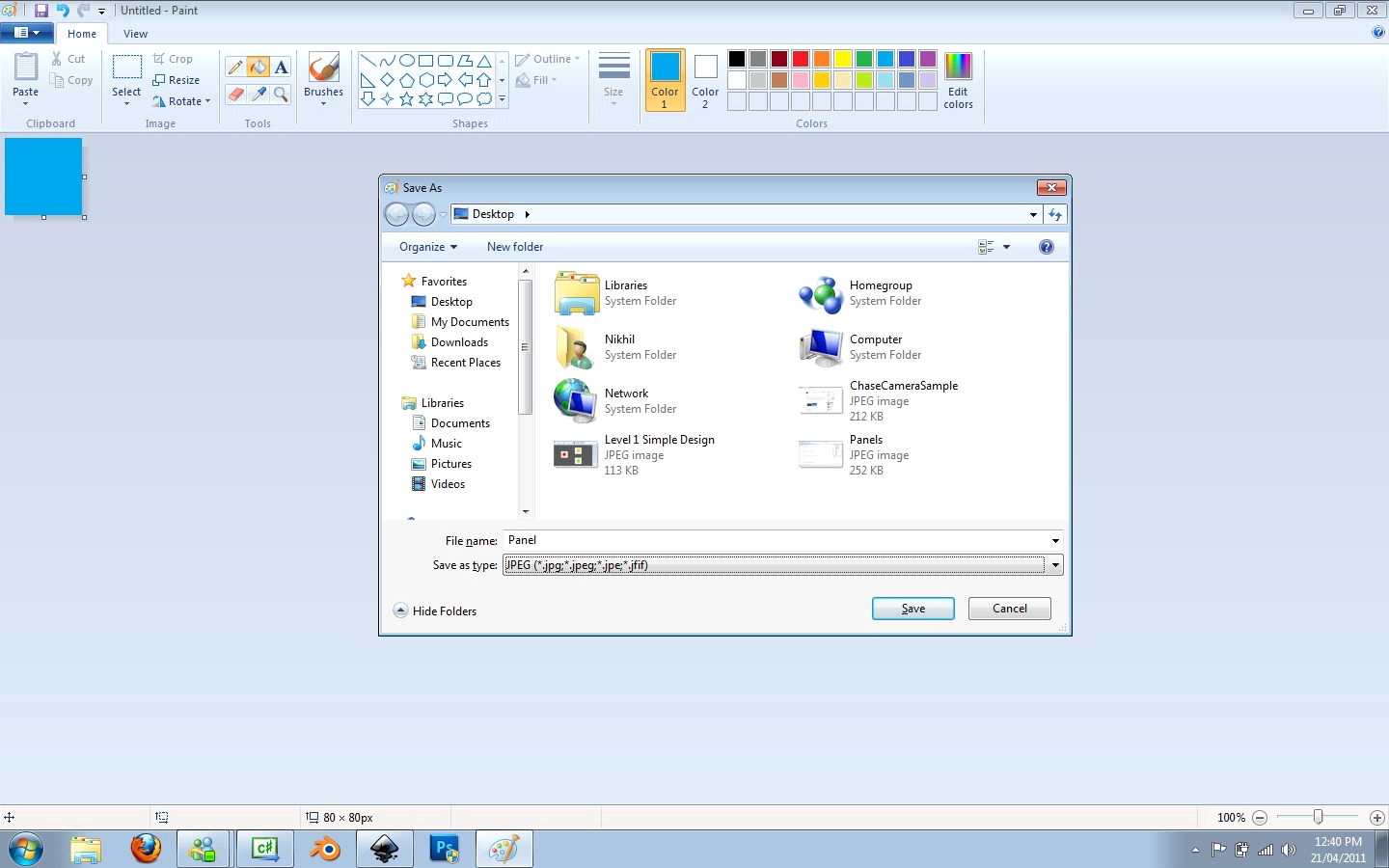
**Prototyping a level design**

Every level starts with a map. I've still not got a clear picture of what a typical level is going to look and play like, so I drew up a number of prototype level blueprints, with objects and characters I had in mind. All these were done in Inkscape/Photoshop and colored contrastingly to differentiate the features of each level. This also speeds up Collision Detection via image pixel colour data.

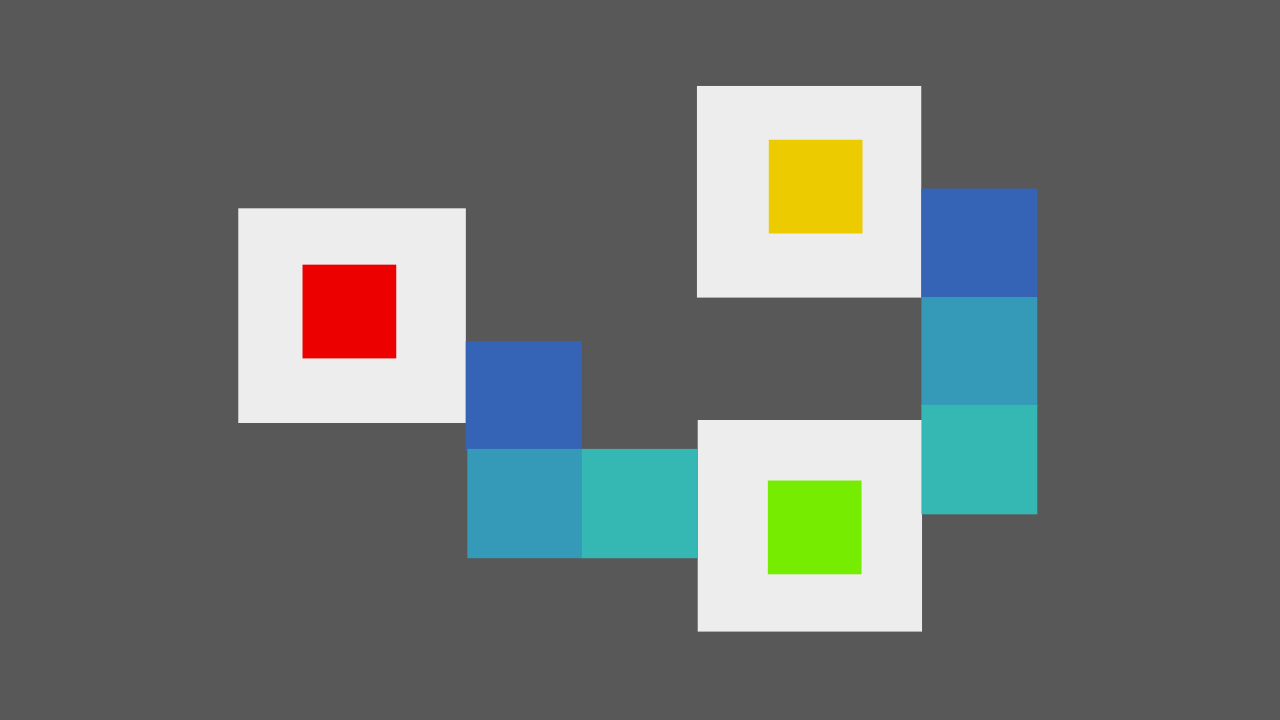


**Panels: Floors of Tomorrow**

Eventually, I decided on a new, interesting gameplay mechanic: Panels. Panels were introduced in the teaser trailers for Portal 2 as "walls of tomorrow", but in this game, they act as dynamic floors, activated by movement and object placement. I made a quick image in Paint.



The first level design consists of alternating panels that activate more panels as the players step on them.



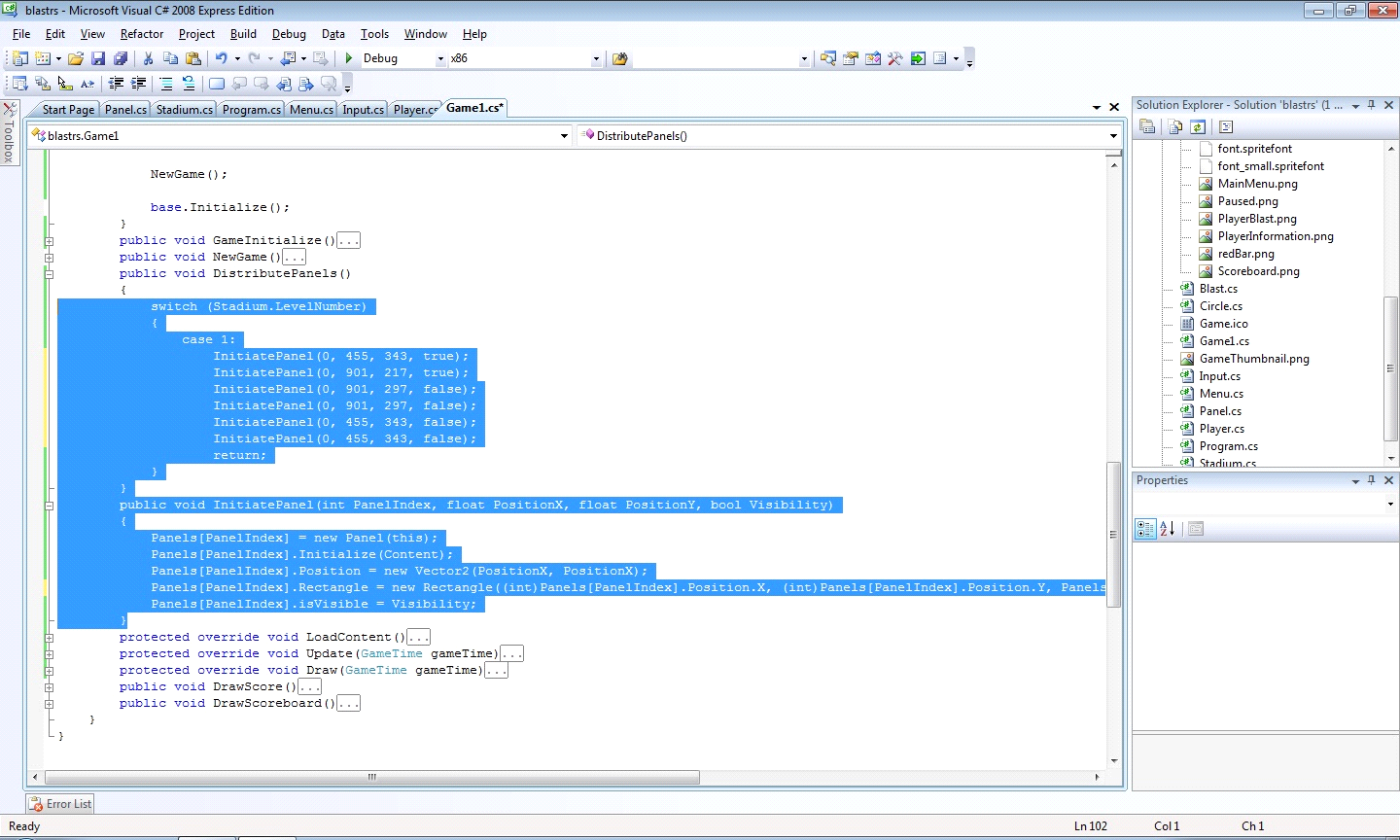
As player 1 steps on a panel, the corresponding panel for player 2 becomes visible, until both players reach the green square and end the level.

**Panel, Appear!**

Once I imported the level texture, I had to work out a method for generating panels and making players walk on them.

I created a Sub that set pre-determined values for each Panel, and I simply call the Sub with the values that I want the new Panel to assume. So, for a new Panel, I simply add:

Panel1.InitiatePanel (thisGame, PositionX, PositionY, Visib le);



This speeds up Panel generation and makes it somewhat dynamic. I can use this method for other objects too, if required.

The panel does not exist on the collisionMap, so it is not detected by the traditional collision code. This means that specialised code must be written to handle player movements on Panels in a static fashion as Panel behaviours upon player interaction can vary across levels. In the next levels, I don't want the panels to activate more panels.

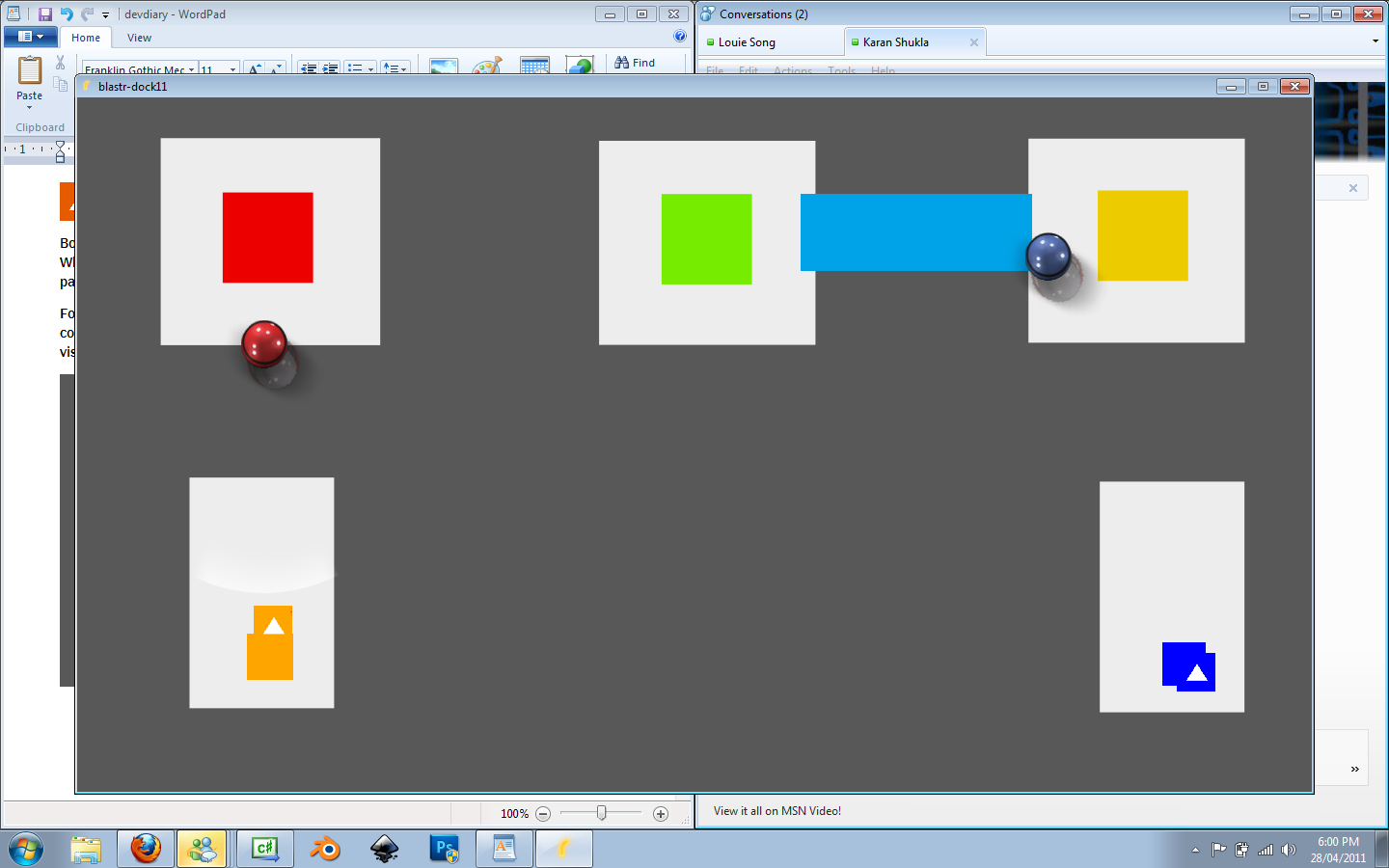
**Level 2: Boxes**

I wanted to introduce a new gameplay mechanic as soon as possible in the fear that I may forget about it, so I decided to leave panels aside for the moment and created Boxes.

Boxes are inanimate objects that are 'unlocked' when placed on top of special surfaces. When 'unlocked', the level can present a number of different scenarios such as visible panels.

For level 2, I wanted to start with the basics, so I created 2 different-coloured boxes and 2 corresponding surfaces. Both players have to move the Boxes onto the surfaces to make visible the Panels to the exit.



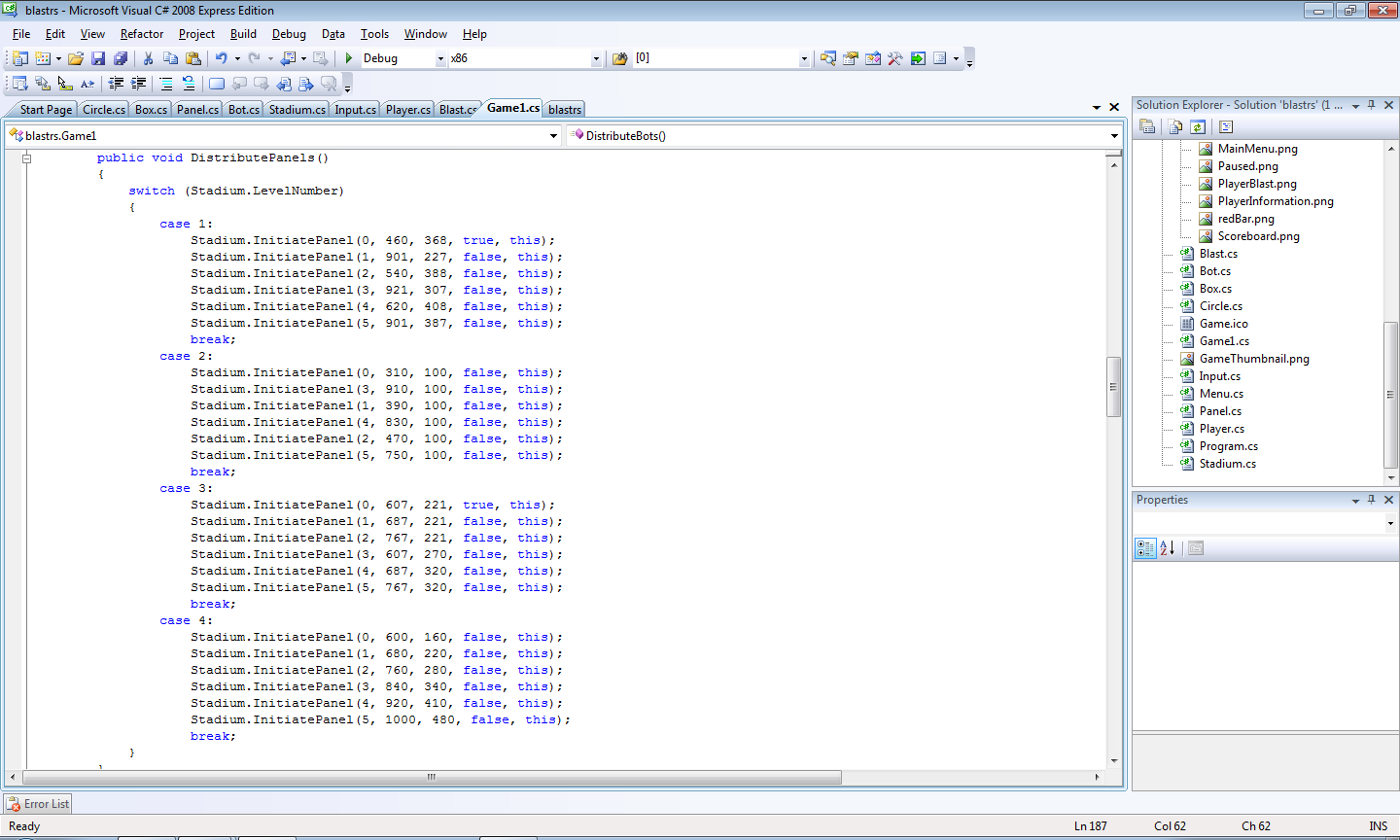
**Level 3: Co-operation Time, with Panels**

I've got boxes working pretty well, so I shall focus my energies on developing levels based on the Panels mechanic. Most of these levels will be based on teamwork and co-operation. These were the ideals behind level 3.



In this challenge, stepping on one of the panels causes all the others to disappear, leaving one of the players stranded while the other must activate the panels and bring them to the exit.

Panel generation code, being dynamically generated using the Initiate Sub, was simple. The panels for this level were placed manually, by reading the X and Y values from Inkscape and passing it into the Sub. This has, to date, created a long 'wall' of code.

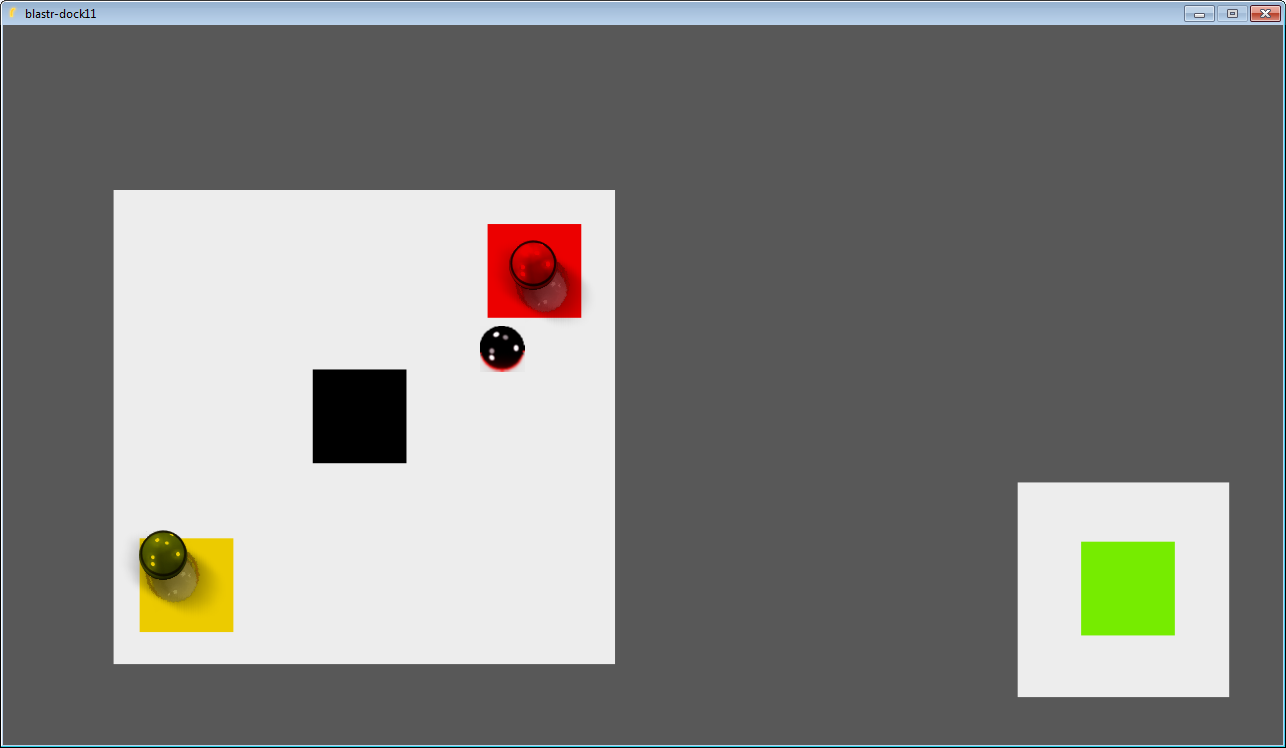


This could be avoided by reading values from an XML file, but I want to get prototyping quickly and there aren't many gains from doing such a thing.

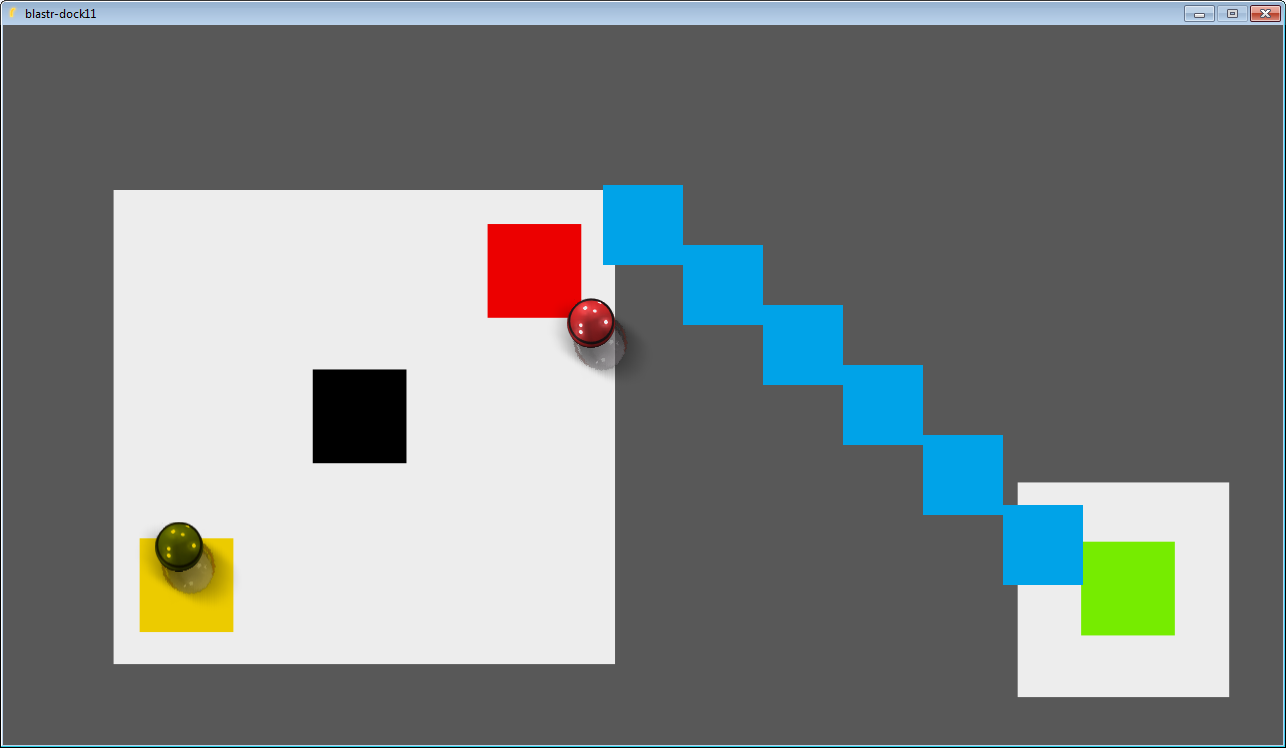
* **In-code**: Quicker- Needs no implementation, Messier, Confusing
* **XML**: Slower- Have to build code to interface with it, Cleaner, Flexible

**Level 4: Bots?**

A friend of mine suggested the notion of building 'bots', essentially dumb AI that prompts a sense of urgency in puzzle solving. I decided to build a level testing the strengths and weaknesses of including an AI within this puzzle-based environment.



When the bot is blasted out of the surface, it triggers a pathway of panels that leads to the exit.

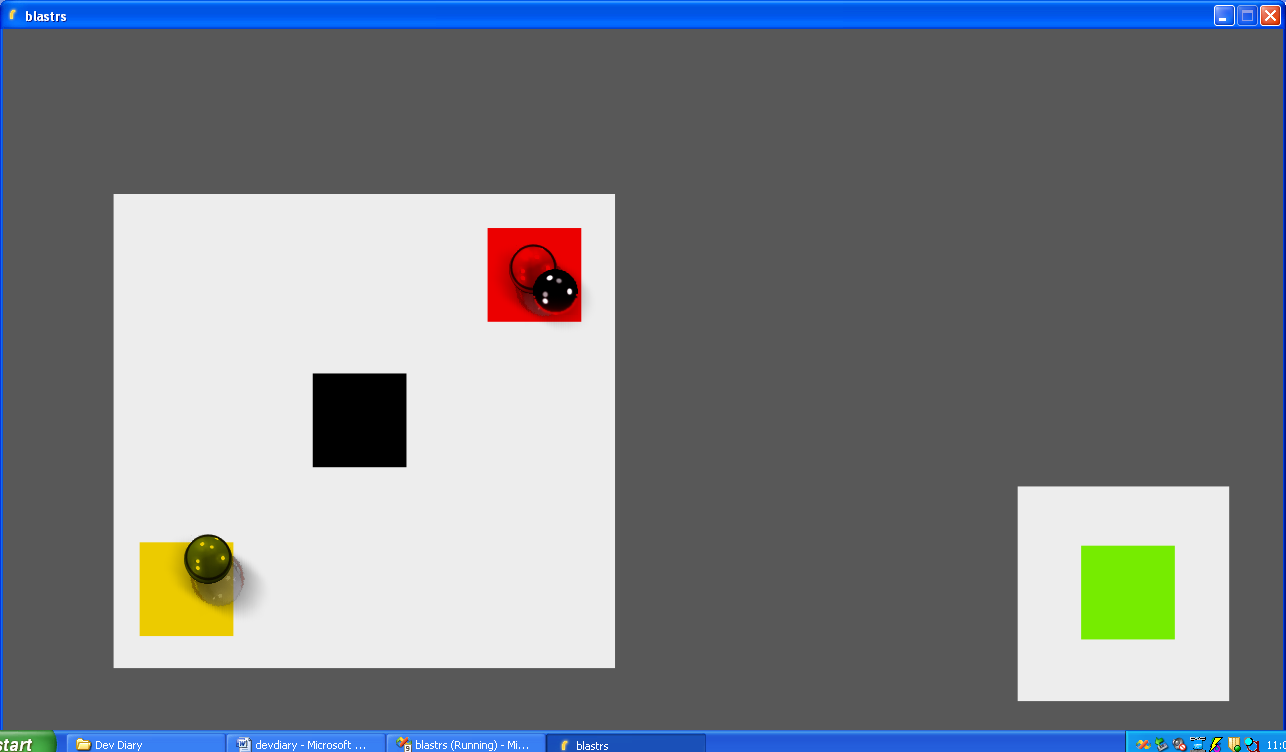


This test did not work as planned:

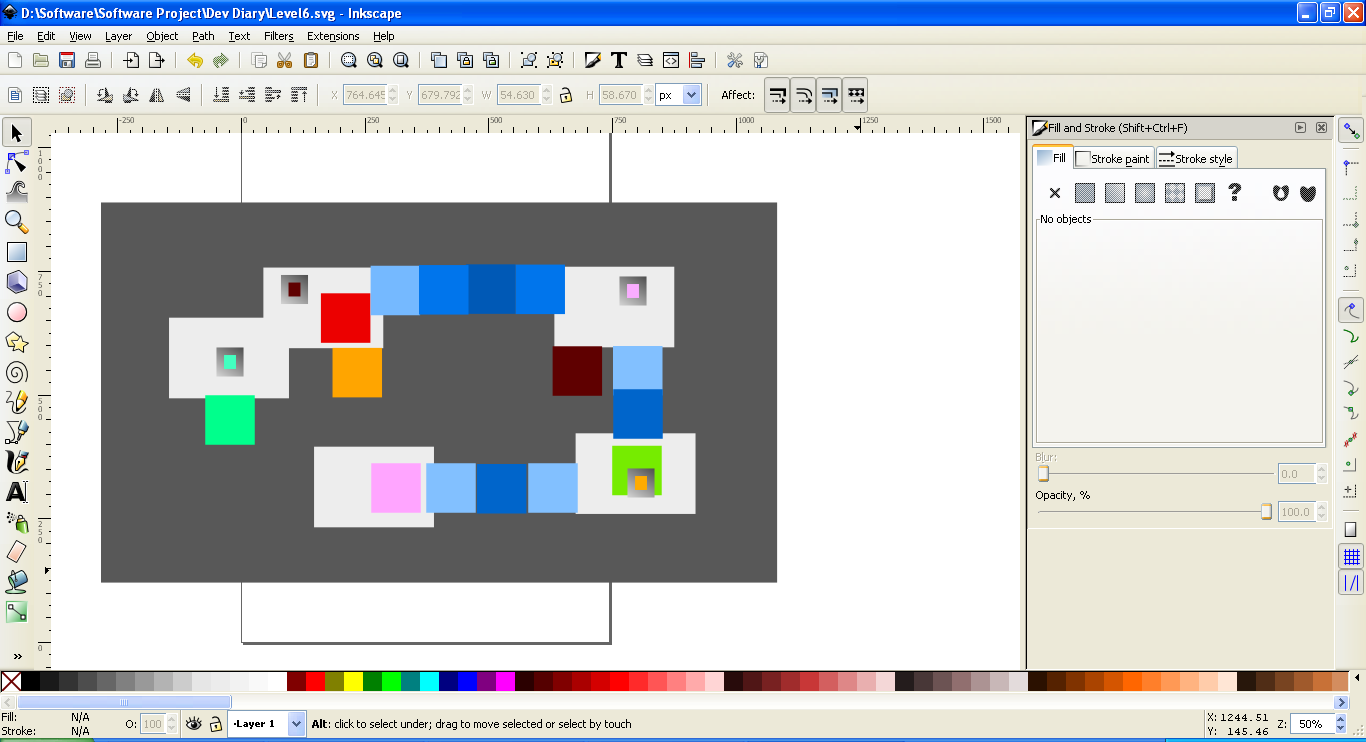
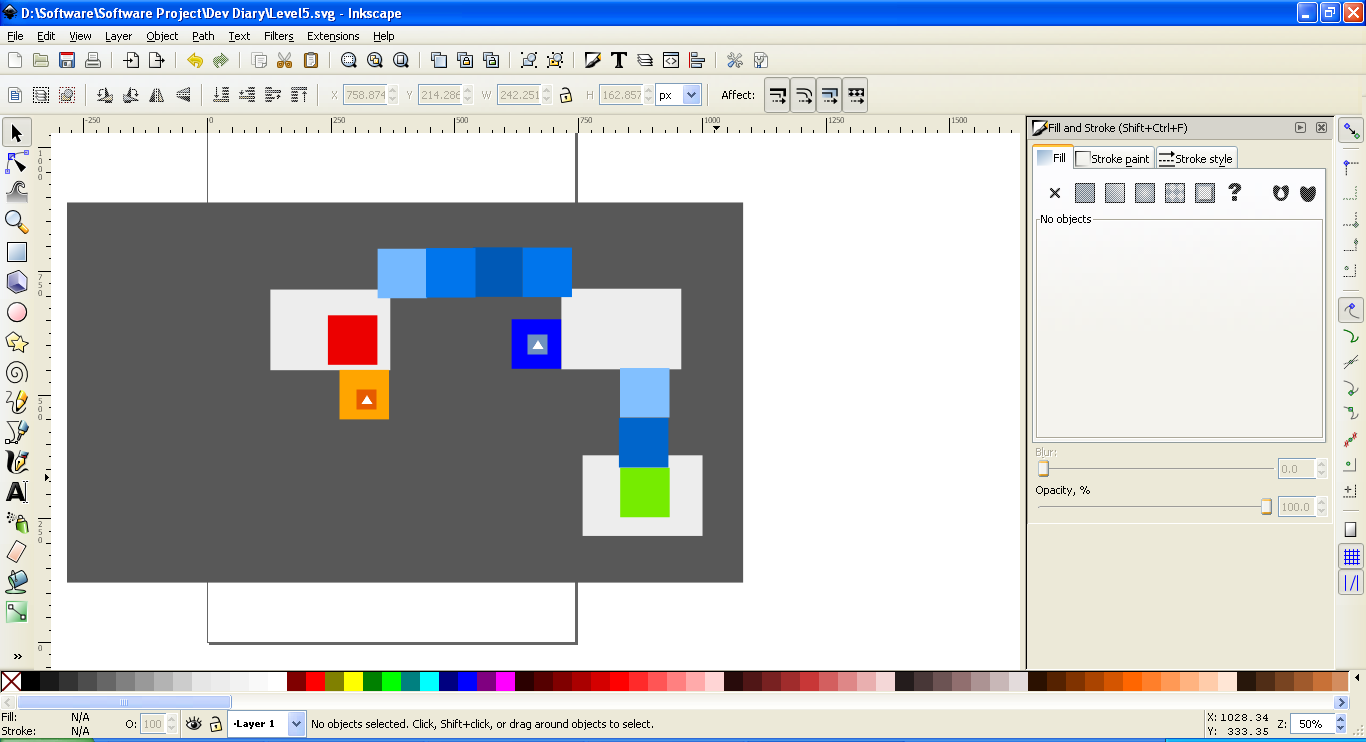
* The bot was too tame; it did not pose a real threat to the players
* Committing suicide by moving off the surface induced the bot to do the same, detroying gameplay
* AI is an incredibly detailed branch of game development. There are developers with special teams simply for developing AI. Thus, a real AI bot will prove to be hard to develop within the project time-frame.

I'm keeping bots on the backburner; I already have a couple of good gameplay mechanics that should suffice for this project.

Also: The Player Selection Algorithm I detailed earlier was of great use in the programming of the AI, to help select which player was closer to the bot.



**Levels 5 and 6: Back to Boxes and Basics**

I decided to go back and add a few more complex levels that take advantage of the Boxes and Panels.  These levels require more thinking on the part of the player as boxes are placed in difficult to access areas.

Also, I made these 2 levels purely singleplayer only. This meant a few changes to the code, disabling player 2.