Diamond Crush Development Plan

- Biweekly iterations with different customer stories
- Task-oriented Test Driven Development
- Context-adapted Extreme Programming

Topics to fix in mind during the works:

During the project development, it's reasonable to keep fixed in our minds some basic guide elements to follow:

- **Portability**: the code must be designed to work on every platform;
- **Compatibility**: we need to avoid conflicts and compatibility issues, and to minimize the system requirements;
- **Accessibility:** the gameplay mechanics must be simple (easy to learn and difficult to master), and the learning curve well balanced;
- **Competitivity**: we need to boost the competition between the two players, to improve their interest and motivation.

This makes necessary the adoption of some self-imposed rules:

- ✓ Never race while coding: better working with calm and attention, to avoid bigger time losses later;
- ✓ Don't make simple things difficult: we can always find an easier way to do what we need to;
- ✓ NEVER add elements that weren't expressly requested in the story or tasks: we aren't gonna need them, and we'll discuss them later;
- ✓ Communicate as much as we can: if there's anything obscure, we need to discuss even the least important details, to make everything clear.

Detailed development plan:

- **Iteration 1** (09/19/2005 - 10/02/2005) **COMPLETE**

  Story 1: The game must draw a window with a diamond on a black background, and then play a sound.
  Task list:
  - **1.1.1**: Open a fixed-size window (with an 800x600 pixels wide client area) - **cdimauro**
  - **1.1.2**: Initialize OpenGL for 2D graphics and color the background in black via OpenGL - **TigerShark**
  - **1.1.3**: Initialize OpenAL - **cisc**
  - **1.1.4**: Load the diamond picture in a texture - **Vifani**
  - **1.1.5**: Draw the diamond texture over two triangles who form a square in the middle of the window - **Vifani**
  - **1.1.6**: Load a sound and then play it - **cisc**
  - **1.1.7**: Draw a 64x64px picture for the diamond - **Antares88**

  Story 2: The diamond can be moved around the screen by using the four directional keys. Every time it collides with any border, its movement in that direction is stopped, and the game plays a sound.
  Task list:
  - **1.2.1**: Allow the diamond to be drawn everywhere in the screen - **Vifani**
  - **1.2.2**: Allow the diamond to be moved by X pixels in the four main directions - **Vifani**
  - **1.2.3**: Detect any keypress for the directional arrows, and consequently move the diamond - **cdimauro**
  - **1.2.4**: Detect collisions between the diamond and the window borders, and then stop its movement - **DanieleC88**
  - **1.2.5**: Play the sound while the diamond collides with any border - **BlueDragon**
  - **1.2.6**: Allow the diamond to pulse during its movement - **VICIUS**

- **Iteration 2** (10/03/2005 - 10/16/2005) **COMPLETE**

Story 1: The diamond must be reduced in size up to 32x32px, it must continuously fall towards the bottom of the screen with fixed speed, and the keypresses for upper directions (including the up-left and up-right diagonals) must be ignored. The play area must also be bounded to a 256x448px rectangle, collisions with the left and right borders must not play any sound, and these must stop the diamond's horizontal movement (to prevent it from exiting the play area), but not its vertical one (stopped only by a collision with the bottom).

Task list:

2.1.1: Reduce GEM_SIZE up to 32px - **DanieleC88**
2.1.2: Disallow the sprites to be moved towards upper directions (including diagonals) - **TigerShark**
2.1.3: Bound the sprite movements in a rectangular-shaped 256x448px area - **71104**
2.1.4: Play the sound only when the diamond collides with the bottom of the play area - **TigerShark**
2.1.5: Add a continuous fall movement with fixed speed for the sprite, stopped only by a collision with the bottom of the play area - **cisc**

Story 2: Introduction of a grid (14 rows x 8 columns) with 32x32px cells, who will cover the play area. The diamond must move along the grid, in the allowed four main directions only, and without forgetting the borders. The movement speed must be reduced to the 80% of the actual one, and the diamond must move instantly between one cell and another in the horizontal directions, while its vertical movement must be slower and more fluid.

Task list:

2.2.1: Create a 32x32px texture as a background for the cells - **Jocchan**
2.2.2: Draw the cells inside the play area represented by extents (8 columns x 14 rows = 112 cells) - **71104**
2.2.3: Create a new play area for player 2, and draw the cells there too - **71104**
2.2.4: Change the horizontal movement for the gems. While the left and right directional buttons are pressed, the sprite must move instantly from one column to the adjacent ones - **Vifani**
2.2.5: Create an 800x600 texture as the window background - **Jocchan**
2.2.6: Draw the BG from task 2.2.5 under the play areas - **71104**
2.2.7: Create a transparent logo, to be shown over the BG - **Jocchan**
2.2.8: Draw the logo from task 2.2.7 down in the middle, between the two play areas - **cdimauro**
2.2.9: Split the playarea class into two or more subclasses - **71104**
2.2.10: Code and Test classes auditing. Add tests for every non-tested line. Eliminate the tests now useless, and update the older ones – **Tutti**

NOTE: This story has been reverted and postponed to the third iteration.

- **Iteration 3** (10/17/2005 - 10/30/2005) **COMPLETE**

  Story 1: The coordinate system used by the game is defined with the origin of both axis on the top-left corner of the screen. We need to translate these coordinates, to have a compatibility with the format used by OpenGL.
  A sprite is defined by a rectangular-shaped portion of texels with any size, located inside a texture with any size proportional to the power of two. The sprite hot spot must be located on the top-left corner of the sprite itself.
  Task list:
  - **3.1.1**: Transition to the new coordinate system for all the project and tests - **TigerShark**
  - **3.1.2**: Allow the Texture and Sprite classes to manage pictures with size different from powers of two - **cisc + fek**
  - **3.1.3**: Transition for the hot spot to the top-left corner of the sprite - **71104**

  Story 2: Introduction of a grid (14 rows x 8 columns) with 32x32px cells, who will cover the play area. The diamond must move along the grid, in the allowed four main directions only, and without forgetting the borders. The diamond will move instantly between one cell and another in horizontal directions, while vertically its movement must be slower and more fluid. The play area background is a unique 256x448px texture.
  Task list:
  - **3.2.1**: Create a 256x448px transparent texture as the play area BG - **Jocchan**
  - **3.2.2**: Create a PlayAreaBackground class who draws the grid by using the texture from task 3.2.1 - **Gammaglobulino**
  - **3.2.3**: Create a MxN sprite matrix, aligned to the background, and draw the diamonds inside - **71104 + fek**
  - **3.2.4**: Insert a diamond on the top of the fifth column. Add the alpha channel feature to the Texture class - **Vifani**
  - **3.2.5**: While the left/right directional keys are pressed, the diamond must instantly move to the adjacent column - **fpitt**
  - **3.2.6**: Move the gravity and collision management to the Grid class - **fpitt + cisc**

- **Iteration 4** (10/31/2005 - 11/13/2005) **COMPLETE**

  Story 1: Introduction of a fixed temporization for the movement of the diamond in the play area.
  A single keypress of the left and right directional arrows must generate a single-column movement. While these keys are kept pressed, the gem will keep moving in the corresponding direction once per fiftieth of a second, unless any collision with the lateral borders verifies. While the left and right directional arrows are kept pressed at the same time, the gem won't move in neither direction, and will keep falling down (unless a collision with the lower border verifies).
  Secondly, the game engine will draw a fixed .jpg image, 800x600px wide and with 24-bit color depth, as the main window background.
  Task list:
  - **4.1.1**: Draw the 800x600px .jpg background picture at the (0,0) coordinates, behind everything - **cover**
  - **4.1.2**: Create a timer who measures time with at least a 5 thousandth of a second accuracy - **BlueDragon**
  - **4.1.3**: Synchronize the rendering of each frame to 1/50 seconds, adding a loop that waits the 20 milliseconds needed to draw a frame before leaving the control - **71104**
  - **4.1.4**: Input must ignore consecutive presses of the same key if the timeout between any keypress is less than 1/50 seconds, considering the first keypress only - **TigerShark**
  - **4.1.5**: Refactoring. Allow Drawable to define in which layer the objects will be drawn - **VICIUS**
  - **4.1.6**: Refactoring of the Game class. Move all the creation process to the class constructor, and create a working gameLoop - **cionci**

  Story 2: Introduction of the down directional key. While this key is pressed, the gem must increase its falling speed by a fixed constant value. While the user releases the key, the falling speed will get back to the usual one. Everything must be implemented without forgetting the possible keypress of both left and right buttons at the same time (the possible combinations will become left+right+down, left+down, right+down). The game must react to any command instantly, catching many consecutive keypresses and without having undesired multiple gem movements while a key is pressed just once, for a slightly longer time than usual.
  Task list:
  - **4.2.1**: While the “down” key is kept pressed, the gravity value used by Grid must be multiplied by a value defined in GameConfig. When the user releases the key, the value must get back to the original one - **cionci**
  - **4.2.2**: Record in a queue inside Input the “press” and “release” events for the various keys. A timestamp must be associated to each event - **71104 + cdimauro**
  - **4.2.3**: Modify reactToInput to use the queue from task 4.2.2 in Input. The actual behaviour must remain the same as now - **71104**

4.2.4: After reacting to a command, reactToInput must remember just one more command during the consequent delay, and must execute it after its end - **POSTPONED**

- **Iteration 5** (11/14/2005 - 11/27/2005) **COMPLETE**

Story 1: Service story, dedicated to an improvement of the response to the user inputs. The response must be immediate and precise, without the loss of any command, even in the wildest situations. Again, we must be sure we won't have undesired multiple gem movements while a key is pressed just once for a slightly longer time than usual, and we need to correct the placement bug of the gem itself while the left and right buttons are pressed at the same time.
Task list:

5.1.1: Implement BlueDragon's Input management algorithm from the trunk/ branch - **71104 + BlueDragon**

Story 2: Creation of a new gem while the previous one collides with the ground or another gem below (the sound must be played in both cases), with a collision detection between the gems located inside the play area (moving a gem to a certain cell will be impossible if it's already taken by another one). Introduction of a counter for the player's score, initialized to zero, with 8 digits values and shown inside the left Score box.
Task list:

5.2.1: While the gem collides with the ground, another one must be created on top. All inputs will have effect on this one only - **cionci**
5.2.2: Modify Grid to stop the gem movement if the cell below is busy (pair programming only) - **cover + TigerShark**
5.2.3: Stop the gem lateral movement if the destination cell in the next column is busy - **DanieleC88**
5.2.4: Allow the game to draw an eight-digit integer number, by using the Sprite class and a texture - **71104**
5.2.5: Show in the left Score box with coordinates (219, 421) the number of the gems inside the play area. The hot spots for each digit must differ horizontally by 11px - **Vifani**

- **Iteration 6** (11/28/2005 - 12/11/2005) **COMPLETE**

  Story 1: Each gem must have a color (out of the 5 available: white, red, blue, green, yellow), a different png (respectively diamond, ruby, sapphire, emerald, topaz) and a different score value (for now respectively 1000, 500, 600, 400, 800), and must identify if, in the adjacent cells (in the four main directions only), there are other gems of the same type.
  The color of the first six gems is randomly chosen at the beginning of the game, and then added to a queue. Every time a new gem is created, the first value is extracted from the queue, and another random one is enqueued.
  Task list:
  - **6.1.1**: Allow the game to choose which type of gem will be created by GameGemFactory when Create is called. The available gem types are: diamond, ruby, sapphire, emerald, topaz - **71104**
  - **6.1.2**: The Score Box must not show the number of gems in the play area anymore. Its new value will be the sum of the values represented by each gem. The gems listed in task 6.1.1 will have these values: 100, 50, 60, 40, 80 - **TigerShark**
  - **6.1.3**: Add a method to change a sprite's brightness - **Ufo13**
  - **6.1.4**: Grid must brighten the gems who have at least one side in common with another one of the same type - **71104 + DanieleC88**
  - **6.1.5**: Add to GameGemFactory a queue who records the next 6 gems to create - **cionci**
  - **6.1.6**: While a new gem is requested, GameGemFactory must return the first one in the queue, and eliminate it from the list. Then, another one must be created and enqueued - **Ufo13**

  Story 2: Each gem must have a continuous animation, to simulate reflections, whose cycle repeats every 3.5 seconds, a constant value kept in GameConfig. The animations of every gem on screen must be synchronized, and must start all together at the same time.
  The "next" box will show the two gems who follow the falling one in the queue. The lower one will be the next one to fall, and this pair will be updated every time a new gem is created.
  Task list:
  - **6.2.1**: Show in the "next" box the next two gems to fall. The lower one will be the next to be released - **71104**
  - **6.2.2**: Allow Sprite to load textures with different animation frames - **TheBol**
  - **6.2.3**: Implement the animations in sprite, switching frames every tenth of a second. After the animation, the base frame must be shown for 3500 milliseconds before restarting the cycle - **cionci + fek**
  - **6.2.4**: Synchronize the animations of the gems inside the play area. If one gem is created while the other ones are in the midst of the animation, it must be in the midst of the animation too - **cionci + fek**

- **Xmas Iteration** (12/12/2005 - 01/06/2006) **COMPLETE**

Story 1: Service story. While the whole central column in which the gems are created is full, the game must stop creating them, and must show a Game Over png in the middle of the play area. We must eliminate the uncertainty that verifies if the user presses a directional button (left or right) while the other one is already pressed. In that case, the gem must remain in the column it was located before the contrasting keypress.
Task list:
Xmas.1.1: Stop inserting new gems while the central column is full, and then show a “Game Over” picture inside the grid - **71104**
Xmas.1.2: Add a 300 milliseconds delay between the moment the falling gem stops and the creation of the next one - **cisc**
Xmas.1.3: Ignore the keypress of a directional button while the opposite one is kept pressed - **cionci**
Xmas.1.4: Change to 50 milliseconds the repeat delay after it passes by once. Revert it back to its original value when the key is released - **Ufo13**

Story 2: The gems must fall in pairs, even if they both fall independently. The one created below is the pivot gem, the ther one will be the slave. By pressing the Z key, the slave must rotate (without any picture change) 90 degrees anticlockwise around the pivot. By pressing the X key, it must rotate 90 degrees clockwise. If the destination cell for the slave gem is busy, and the rotation is impossible, it will keep rotating in the same way until a free cell is found. The rotation effects are as described here:

- Slave over the pivot -> the slave will go respectively on the left or right of the pivot
- Slave on the left of the pivot -> it will go respectively under or over
- Slave under the pivot -> the slave will go respectively on the right or left of the pivot
- Slave on the right of the pivot -> it will go respectively over or under

By pressing the C key, the slave will rotate 180 degrees.
If one gem collides with any object below, the other one – if there won’t be anything below it – will keep falling independently. The Next box will keep showing, without any uncertainty, the pair of gems who will fall immediately after the actual one (the lower one will be the new pivot, the upper one the new slave).
Task list:
Xmas.2.1: GridControl must create two gems now. The first one, the pivot, will be under the player’s input control. The slave will follow the movements of the pivot - **TheBol + Ufo13**
Xmas.2.2: Detect if the Z key is pressed, and then rotate the slave 90 degrees anticlockwise around the pivot - **DanieleC88**
Xmas.2.3: Detect if the X key is pressed, and then rotate the slave 90 degrees clockwise around the pivot - **VICIUS**
Xmas.2.4: Detect if the C key is pressed, and then rotate the slave 180 degrees around the pivot - **VICIUS**
Xmas.2.5: If one gem collides, the other one loses the player’s control, and falls below - **71104**

- **Iteration 8** (01/09/2006 - 01/22/2006) **COMPLETE**

  Story 1: Introduction of a pulsing system, to highlight the pivot, making it easier to identify. This pulse, unlike the previous one, must be visual only, and must not affect the calculated gem size and collisions in any ways.
  When the destination cell, after a 90 degree rotation, is busy, the game won't repeat the rotation anymore. The command will simply be ignored.
  We also need to split the rotation delay and the lateral movement one to two different values, with 200 and 50 as the respective defaults.
  Task list:
  - **8.1.1**: Allow Sprite to pulse. The pulse must verify in the play area only, and this must not influence collisions nor the calculated gem size. The pulse size can be changed in GameConfig - **VICIUS**
  - **8.1.2**: Highlight the pivot in a GemsPair making it pulse. The pulse must stop when at least one gem in the pair collides - **Bonfo + 71104**
  - **8.1.3**: Disable all the code and tests used to repeat a rotation until a free cell is found. If the destination cell is not available, the gem won't rotate at all - **fek**
  - **8.1.4**: Allow CommandHandler to set any value for the different keys delays - **TigerShark**
  - **8.1.5**: Change MirrorSlaveGemCommandHandler, RotateClockwise-CommandHandler and RotateCounterClockwiseCommand-Handler to give them a bigger FastRepeatDelay than default. Put this value in GameConfig and use 200 as default - **DanieleC88**

  Story 2: While a gem is in touch with others of the same color, and these form a quadrilateral inside the grid (a square or a rectangle of any size), they will melt to an agglomerate. The gems forming the agglomerate will use different pngs according to their positions along the borders (4 sides and 4 angles) or inside it. The agglomerate can be considered as a complex object, which score value is given by the sum of the single gems ones, multiplied by a bonus (the bonus value is unique for all agglomerates, and can be changed in GameConfig). The gems who melted to an agglomerate cannot turn back to single gems anymore.
  After the end of the story, the brightness change for adjacent gems must be deleted.
  Task list:
  - **8.2.1**: Change the code that identifies adjacent gems of the same color to discern only the gems that form a square or a rectangle - **Vifani**
  - **8.2.2**: When a square or a rectangle is discerned, draw Jocchan's tile pictures in place of the gems - **TheBol + ?**
  - **8.2.3**: Allow BagOfGems to multiply the score of the gems contained in an agglomerate by a Bonus. The Bonus value is kept in GameConfig - **?**
  - **8.2.4**: Delete the code and tests that brighten adjacent gems - **?**

Story 3: Refactoring and bugfix service story.
Task list:
- **8.3.1**: Add tests for all classes and function not already covered by 100% - **?**
- **8.3.2**: Refactoring to make the game loop in Game independent from the game initialization, to make GameLoop able to be tested - **?**

- **Iteration 9** (01/23/06 - 02/05/06) **COMPLETE**

Story 1: Introduction of a new droppable object called *Treasure Box* (with the same colors as gems and null score value). While a treasure box collides with the bottom or another gem (or agglomerate), every gem (or agglomerates) with the same color, adjacent to the box or to any other similar gem in any of the four main directions, will be deleted, increasing the player's score with the sum of the score values of all the deleted gems, and any empty space left by these deletions will be filled by the gems above (gem agglomerates will fall ONLY if there won't be any gem left below their whole width).
We'll define this process "deletion", and every time we'll refer to a deletion, we'll refer to this entire process.
Gem-box collisions must be catched even when gems collide with treasure boxes, triggering in the same way the deletion process described above. This can trigger chain reactions (Crushes).
No gem pairs must be created before the end of all deletions.

- **Iteration 10** (02/06/06 – 02/19/06) **COMPLETE**

Story 1: Introduction of a secondary grid, capable to manage the same objects as the already existing one, and controlled by a second player by using the numpad (8, 4, 6 and 2 for the four directions; 7, 9 and 5 for Z, X and C).
The gems queue must be the same for both players. Every time a pair collides, and another one gets enqueued, the same one must be appended to the other player's queue, to keep both queues identical in any situation.
The second Score box must also show the other player's score, and the second Next box must show the next pair to fall in the second grid, in the same way as the first one.

Story 2: Definition of a GameTurn as an integer increased by 1 every 20 milliseconds. Each value expressed in milliseconds in the whole code base must be converted to GameTurns.
Introduction of an event logging system, to save on a txt file every key press or release, together with the corresponding GameTurns.

- **Iteration 11** (02/20/06 - 03/05/06)

  Story 1: Introduction of a counter for Crushes, set to 1 after a first deletion, and then increased by 1 every time that, when the gems above fall down to fill in empty cells, other deletions occur. While this counter has a value bigger than one, different pngs, counting the current Crush's length, will be shown on the left side of the screen for the first player, and on the right one for player 2.
  After the end of the Crush, the score value of all the gems deleted during the Crush itself, before being added to the whole Score, must be multiplied by the value held in the Crush counter. Then, this counter will be set back to zero, and the pairs will start falling again.

  Story 2: Introduction of a counter for the total amount of gems (treasure boxes must not be counted) deleted in a Crush longer than 2, whose value (updated after the end of the Crush itself), if bigger than zero, will be shown in a Warning box below the opponent's play area. In the same way, the value of the opponent's counter will be shown below the player's play area.
  If no deletion occurs, or if the Crush counter has a value lesser than 2, this counter will be set back to zero, and the corresponding Warning box will be hidden.

- **Iteration 12** (03/06/06 - 03/19/06)

  Story 1: Bugfix and refactoring.

  Story 2: Introduction of *Flashing Gems*, special droppable objects with null colors and score values. When a Flashing Gem collides with another droppable (gem or treasure box), every gem and every treasure box with the same color as the touched droppable must be deleted from the play area, without any score nor any effect in any triggered or active Crush.
  The color of the droppables to delete will be decided giving priority (in order) to the objects below, on the left, on the right or over the *Flashing Gem*.
  This droppable must obiouvsly have a very low occurrency percentage, quantifiable as 2% and whose value can be changed in GameConfig, together with the occurrency percentages of the other droppables.

- **Iteration 13** (03/20/06 - 04/02/06)

  Story 1: Introduction of *Stones*, another droppable defined by a color (like gems), a null score value and an integer, chosen between 1 and 5, according to the rows (from the upper to the lower ones) where they will be placed, following the scheme below.
  This number will be shown in the png used by the stone itself, and its value will decrease by 1 every time a pair gets dropped by the player. When this number gets down to zero, the stone will turn into a gem of the same color.

Every time the player drops a pair, and before the next pair starts to fall, some stones, in a number equal to the counter in the Warning box below his play area and with random colors, will fall (with increased speed) in his play area, from the left to the right and eventually on more layers, one above the other. After this process, the counter will get back to zero, and the pairs will start falling again.

```
Stones scheme
Rows        ->  Value
13, 12      ->  5
11, 10      ->  4
9, 8, 7     ->  3
6, 5, 4     ->  2
3, 2, 1, 0 ->  1
```

- **Iteration 14** (04/03/06 - 04/16/06)

  Story 1: Every time the player drops a pair, and before the next pair starts to fall, some stones, in a number equal to the counter in the Warning box below his play area, and with colors decided by following the pattern below, will fall (with increased speed) in his play area, from the left to the right and eventually on more layers, one above the other. After this process, the counter will get back to zero, and the pairs will start falling again.
  Before turning into a gem, a stone will be deleted only after the deletion of an adjacent gem. Its score value will be zero, but it will count for the number of stones to be sent to the opponent.
  If the opponent manages to delete gems before receiving these stones, he will be able to reduce their number by a value equal to the difference between the incoming and the outgoing ones. If his outgoing stones will be more than the incoming ones, it will be the player to receive their difference. This process is called "Counter Attack", and it will be underlined by a "Counter" png, shown in place of the counterattacking player's Warning box.

  Pattern: The pattern is defined by a 8x1 matrix, set to be repeated vertically without any limit, filled with integers between 1 and 5. Before every match, these numbers will be randomly matched with the colors used for gems, and this will define the colors for the stones to be put inside the play area.
  Our current pattern will be: 1 2 2 3 3 4 4 5.

  FIRST PLAYABLE VERSION

Target dates:
First Playable (pre-Alpha): 04/16/2006 – Iteration 14 (released April 24th)
Alpha: 08/06/2006 – Iteration 23
Beta: 09/03/2006 – Iteration 25
Release: 10/15/2006 – Iteration 28