Miller v. Facebook, Inc. et al Doc. 148 Att. 1

I. NEEL CHATTERJEE (STATE BAR NO. 173985)
nchatterjee@orrick.com
JULIO C. AVALOS (STATE BAR NO. 255350)
javalos@orrick.com
MORVARID METANAT (STATE BAR NO. 268228)
mmetanat@orrick.com
ORRICK, HERRINGTON & SUTCLIFFE LLP
1000 Marsh Road
Menlo Park, CA 94025
Telephone: 650-614-7400
Facsimile: 650-614-7401

Attorneys for Defendant
FACEBOOK, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

DANIEL M. MILLER,

Plaintiff,

v.

FACEBOOK, INC. and YAO WEI YEO,

Defendants.

Case No. 3:10-CV-00264 (WHA)

DECLARATION OF DAVID CRANE IN SUPPORT OF FACEBOOK INC.'S MOTION FOR SUMMARY JUDGMENT RE CONTRIBUTORY COPYRIGHT INFRINGEMENT

Date: April 7, 2011
Time: 8:00 A.M.
Court: Courtroom 9, 19th Floor
Judge: Honorable William Alsup

Dockets.Justia.com

I, David Crane, hereby declare under penalty of perjury that:

1. I am an expert in video game design. I make this declaration in support of Facebook's Motion for Summary Judgment. Except as otherwise noted, I have personal knowledge of the facts stated herein and if called as a witness, could and would competently testify thereto.

I. SUMMARY OF ANALYSIS

2. I have reviewed the two games in question, Boomshine and ChainRxn, to determine what similarities exist between the two games and what the nature of those similarities are. After carefully reviewing the games in detail, I have concluded that the only similarities in the two games relate to the concepts and ideas of the games (and thus the procedures). To the extent the games have expression independent of the concepts and procedures, the expression in the two games are totally different.

3. I have also conducted extensive analysis of decompiled source code and found no basis whatsoever for Plaintiff's claim that Mr. Miller's source code was copied during the creation of the game ChainRxn. This finding is important because the nature of the code is very closely tied to the actual images created on the screen. If the expressive elements were in fact similar, I would expect the code to also have very similar elements. They do not.

II. QUALIFICATIONS

4. I have been a pioneer in the video game field since the medium's inception. I was one of the early programmers at Nolan Busnell's Atari, Inc. and developed games for the Atari Video Computer System. While at Atari, I developed a number of programming techniques incorporated in dozens, if not hundreds of video games. In 1979, I co-founded Activision, Inc., the first third-party publisher of video game cartridges. Within three years, Activision grew to over $300 million in value and is now the largest video game publisher in the world with a market capitalization of over $13 billion. I continue to design and program games as my primary focus. In my 34-year career in video games I have designed and programmed over 80 commercial game products generating over $400 million in revenues. I have designed and programmed games on

virtually every video game system invented, from the early days of Atari and Magnavox through to present-day systems such as the iPhone and iPad.

5. I have received many other awards for my work and career. Most recently, I received one of the Academy of Interactive Arts and Sciences lifetime honors: The Pioneer Award, celebrating my foundational and continuing work in the creation and development of the video game business. This singular honor, presented to me in 2010, was the inaugural award in a new category. I was the first to receive this award out of everyone who had ever worked in the video game industry throughout its entire history. Additional awards include Game Designer of the Year (twice), the prestigious 2003 Game Developer Choice Award for contribution to the field, and the Lifetime Achievement Award in Video Games from Classic Gaming Expo. In addition to these personal honors, many of the individual games that I have developed have also received numerous awards.

6. I am a regular speaker and/or panelist at video game industry trade events such as the D.I.C.E. Summit (Design, Innovate, Communicate & Entertain), and GDC (Game Developers Conference. A true and correct copy of my Curriculum Vitae is attached hereto as **Exhibit 1.**

III. TERMINOLOGY

7. In my experience, video games contain both functional and expressive elements. In this declaration I use the terms "game concept", "game play", and "game mechanics"[1]. These terms are understood by anyone well versed in the art of video game design to be synonymous with the game idea and the process by which a player interacts with the game. In my experience, these terms may accurately be called the "functional" elements of a game.

8. Expressive video game elements are those that are not critical to the operation of a game. In other words, swapping out one expressive element for another will not affect game play or strategy. For instance, in this declaration I use the terms "color", "music", and "sound effects".

[1] "Game mechanics are a construct of rules intended to produce an enjoyable game or gameplay." This definition was taken from the website http://en.wikipedia.org/wiki/Game_mechanics. A true and correct copy of this webpage is attached hereto as **Exhibit 2.** This screenshot accurately displays the webpage as I viewed it.

In video game design these terms convey a similar meaning to their use in other fields. As a group these terms are used to represent aspects of the game that are "expressive" in nature.

9. Finally, in this declaration I also use the term Flash. "Adobe Flash (formerly Macromedia Flash) is a multimedia platform used to add animation, video, and interactivity to web pages. Flash is frequently used for advertisements and games."[2] One common use of the term Flash refers to an authoring tool, sold by Adobe Systems either as a standalone product or bundled with its Creative Suite of products. Users of this tool can create interactive graphic content using built-in vector graphic primitive, and/or imported images and sounds, manipulated through the use of a scripting language: Actionscript.[3] A game authored with this tool is referred to as a "Flash game" or as "written in Flash." The two games central to this analysis, ChainRxn and Boomshine, are Flash games.

IV. ANALYSIS

A. Introduction and History of the Chain Reaction Game Genre

10. It is widely accepted within the gaming community that there is a genre of video games known as "Chain Reaction Games"[4]. As the name implies, in a Chain Reaction game the player starts a process which causes a reaction, which in turn causes a similar reaction, until the resulting chain reaction exhausts all possible reactions or until it decays and stops. The most common and historically grounded game play relies upon a simulated explosion on the screen such that the explosion expands and ultimately contracts as would a real-world explosion. The explosion's expansion radius is a visual indication of its destructive path, representing the fireball surrounding the explosion. As shown below, this sort of game play has existed for well over 30 years.

[2] This definition is taken from http://en.wikipedia.org/wiki/Adobe_Flash A true and correct copy of this webpage is attached hereto as **Exhibit 3.** This screenshot accurately displays the webpage as I viewed it.

[3] This definition is taken from http://en.wikipedia.org/wiki/Actionscript A true and correct copy of this webpage is attached hereto as **Exhibit 4.** This screenshot accurately displays the webpage as I viewed it.

[4] When I performed a Google search for the term "Chain Reaction Game," approximately 495,000 results returned.

11. If this fireball comes into contact with any other volatile objects, each of those objects will detonate in turn, initiating their own expanding and contracting explosion effect. If any detonated object's fireball contacts any other object, that latter object will explode as well. This game play mechanic allows for a chain reaction that can theoretically explode every object on the screen.

12. The two games at issue in this litigation, Boomshine and ChainRxn, both belong to the Chain Reaction game genre.

1. Historical Development of Chain Reaction Games.

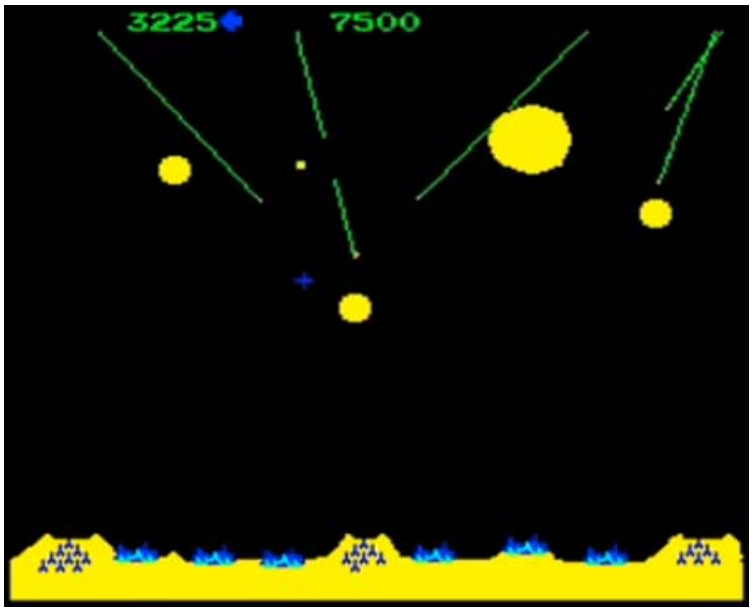


Figure 1: Screen capture of Atari's Missile Command game during play

13. The Chain Reaction game genre has been around since the 1980s. In 1980 the Atari Corporation introduced the arcade game "Missile Command". "Missile Command" embodied a game idea new to the field of video gaming in which the player set into motion a chain reaction designed to blow up as many enemy objects as possible with each targeted blast. The player was faced with a barrage of enemy missiles and had a limited number of friendly missiles at his disposal. To destroy this threat the player would:

(a) Identify the trajectory of the incoming missiles, looking for patterns

(b) Move his cursor to a point on the screen at which to detonate a "friendly" missile

(c) Click a button to cause the explosion at the location of a cursor

(d) Watch as enemy missiles are engulfed by (and contribute to) the chain reaction

(e) Repeat to eliminate enemy missiles that passed outside of the radius of any explosion

This game idea was wrapped inside a Cold War era nuclear defense scenario where the player was firing missiles to create the player-directed chain reaction triggering explosion, and defending cities from incoming missiles. Enemy missiles came in waves with more targets appearing with each successive level.

14. "Missile Command" established the basic Chain Reaction game play:

(a) Use a cursor to select a point on the game playfield

(b) Click to create an explosion at that point

(c) Watch as a chain reaction of explosions destroys on-screen objects

15. "Missile Command" is clearly the prototype for all Chain Reaction games that followed.

16. A Japanese game named "Chaos Theory" represents another example of a game in the Chain Reaction genre. Chaos Theory is somewhat unusual in that it represents a devolution from a story-rich implementation to a stylized simplification. In a sense, the game is purely about the game mechanics with virtually none of the expression. The basic chain reaction style game play from "Missile Command" is still present, with a player-invoked explosion of one ball resulting in a chain reaction of explosions of balls within a certain proximity.

Figure 2: Enhanced screen capture of Chaos Theory during play.
Motion blur added to represent ball object direction and speed.

The game play objects have been reduced to primitive balls or dots, and the goal is to simply explode as many as possible. The balls are launched upward from below the bottom of the screen, and they travel as if affected by gravity – eventually falling off the bottom of the screen if not exploded.

17. Another recent example in the Chain Reaction Game genre is called "Every Extend." This game came out of Japan in 2004[5]. In this game objects move in all directions and the player moves a cursor and clicks to initiate the chain reaction.

[5] Release date April 25, 2004: http://www.giantbomb.com/kanta-matsuhisa/72-91159/ **Exhibit 5**

Figure 3: Screen Captures of "Every Extend" showing groups of rotating cubes in a chain reaction explosion

18. To maximize score the player needs to mentally track the paths of all the objects and wait for a confluence of objects before initiating the explosion. As the first game in the genre to allow objects to move in all directions of the compass, this game rewards gamers who successfully develop the necessary pattern-tracking skills.

19. Making permutations of this sort of game play is common and extended. I personally conducted a review of a number of video game review sites and blogs. I found a number of references describing "Every Extend" as the prototype from which Boomshine was created. One such review concluded: "Danny Miller's Boomshine is a new riff on the chain-reaction action pioneered by Omega's Every Extend."[6] Another advised: "if you're familiar with Every Extend Extra for the PSP you'd like this simplistic version."[7] And a third explained the game as: "Think Missile Command (or more recently Every Extend) without any cities to defend"[8]. And, in fact, in one game programming forum a contributor finds it "…kind of silly that the author is so uptight about 'clones', yet nowhere in the game's site or description does he mention Every Extend as an obvious inspiration?"[9]

20. "Pretty Pretty Bang Bang", seen below in Figure 1, represents another example of a popular Chain Reaction game predating Plaintiff's. The game is featured on the Flash site kongregate.com (a leading browser-based game site with over 13 million monthly unique visitors

[6] This quote is taken from the website located at, http://jayisgames.com/tag/chainreaction. I personally reviewed that website. Attached hereto as **Exhibit 6** is a true and correct copy of a screenshot of that page as it appeared to me through my web browser.
[7] This quote is taken from the website located at, (http://fogdogg.net/digg/Boomshine.swf): http://jetset.jesterball.com/viewtopic.php?t=954&view=previous&sid=dea0b01d1bb506c906490d37d601341f. I personally reviewed this website. Attached hereto as **Exhibit 7** is a true and correct copy of a screenshot of that page as it appeared to me through my web browser.
[8] This quote is taken from the website located at, http://casualtygamer.com/2008/08/flash-game-of-the-day-boomshine/ I personally reviewed that website. Attached hereto as **Exhibit 8** is a true and correct copy of a screenshot of that page as it appeared to me through my web browser.
[9] This quote is taken from the website located at: http://www.gp32x.com/board/index.php?/topic/36312-boomshine-web-game/ I personally reviewed that website. Attached hereto as **Exhibit 9** is a true and correct copy of a screenshot of that page as it appeared to me through my web browser.

who spend more than 23 million hours playing per month).[10] According to that website, "Pretty Pretty Bang Bang" was first published at kongregate.com on October 30, 2006.[11] Object motion is 360 degrees, and one click starts the chain reaction.

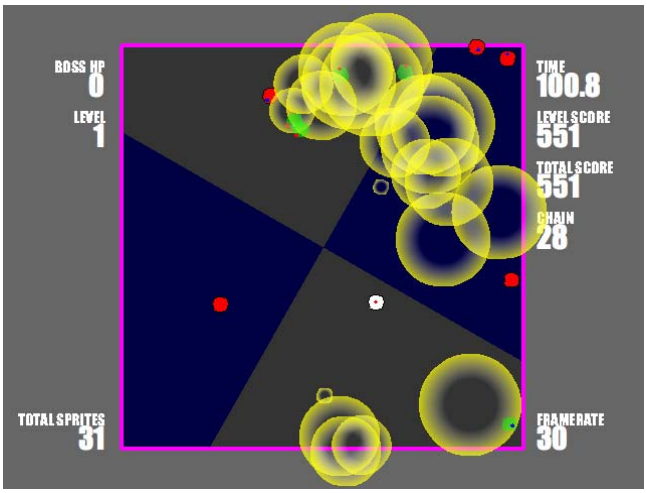
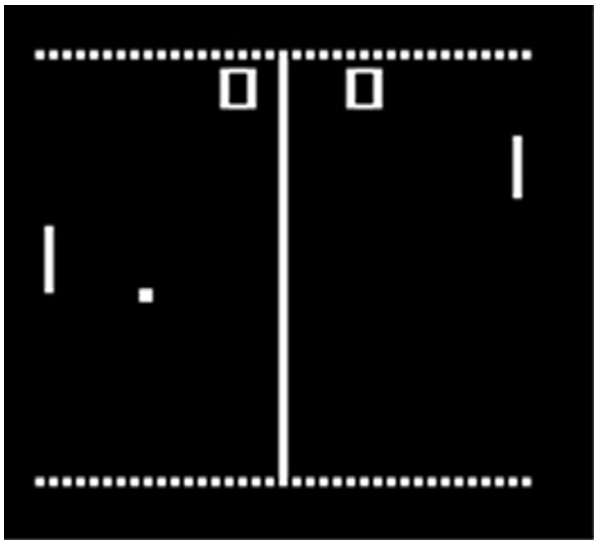


Figure 4: Screen capture of Pretty Pretty Bang Bang

21. The final historical reference is the video game Pong, one of the first video games in history.



[10] I have reviewed a transcript of the February 7, 2011 deposition of Plaintiff Daniel Miller. A true and correct copy of excerpts from that deposition is attached hereto as **Exhibit 10**. Plaintiff states that he used and visited Kongregate.com extensively. *See* **Exhibit 10** [Miller Depo Transcript Excerpts] at 57:11-58:1; 48:1-22; 283:18-284:3.

[11] I viewed this publish date on the website located at: http://www.kongregate.com/games/luvcraft/pretty-pretty-bang-bang. A true and correct screenshot of that website is located at **Exhibit 11**. This screenshot accurately displays the webpage as it appeared to me through my web browser.

Figure 5: Pong, one of the first video games in history

22. Pong was developed nearly forty years ago and featured balls bouncing off the edges of the play screen.

2. Gameplay Features of the Chain Reaction Game Genre

23. In my personal experience, the key to a good game is the ability to improve one's result through the application of a learnable skill to the procedures of the particular game.

24. The game mechanics for games in the Chain Reaction genre do not provide much in the way of manipulative skill. In fact, the instructions for one such game known as "A Chain Reaction Game v.2" state simply: "Just click!"[12] With little physical skill involved, the challenge in this genre of games comes through pattern prediction. In every case the player is trying to explode the maximum number of targets; and this goal is best achieved by mentally following the pattern formed by the moving objects and predicting the best time and position of an anticipated confluence of objects. A chain reaction begun at that spot and at that time can destroy the most targets. This is entirely a procedure of the game and has no expression whatsoever.

25. For example, the volatile objects in "Missile Command" enter the screen from the top and terminate before leaving the bottom of the screen. Predicting their paths to decide on a good move is therefore quite simple. The objects in "Chaos Theory" enter the screen from the bottom, their ascent decays, and they fall off the bottom of the screen. This requires a different level of pattern prediction – adjusting to two different directions of object motion.

26. New game ideas incorporated into the Chain Reaction genre of games occasionally appear. For example, in both of the aforementioned games "Every Extend" and "Pretty Pretty Bang Bang", the objects move in all directions across the screen. The resulting patterns formed by the moving objects change the game play dramatically. These examples are illustrative of how changing object movement dramatically changes the skill factor of a game in the chain reaction genre.

[12] This quote was taken from the website located at: http://www.addictinggames.com/chainreactionv2.html A true and correct copy of a screenshot of this website is attached hereto as **Exhibit 12**. This screenshot accurately reflects the webpage as it appeared through my web browser.

B. **Comparison of The Boomshine And ChainRxn Games**

1. Versions of the Boomshine and ChainRxn Games Used In My Analysis

27. In my experience, video games published online often end up with a number of different versions available to the public. For this analysis, particular versions of the two games at issue had to be selected. I used the version of the Boomshine game found at http://www.k2xl.com/games/boomshine/ during the months of January and February 2011. I believe this to be Version 1.11 3.9.07. I understand that these are the versions that underlie Plaintiff's infringement allegation.

28. My comparison was performed using the version of the ChainRxn game that is publicly available at http://chainrxn.zwigglers.com.

29. The two games central to this analysis, ChainRxn and Boomshine, share an evolved game concept with respect to object motion. The objects in both games move in multiple, fixed directions. Further, the game play action is constrained to a single screen by having the objects reflect back into the field of play rather than leave the game playfield. This movement deeply affects the game mechanics and strategy. It does not change any expression in the game.

30. This game play concept allows for the movement patterns to be completely predictable, even deterministic. A player skilled in pattern recognition can predict where to start the chain reaction to explode the maximum number of objects. ***Remove this fundamental game play concept and you would have a completely different game.***

2. Boomshine's and ChainRxn's Component Parts.

31. I have personally reviewed the Boomshine and ChainRxn games. This Section contains the substance of that review. In performing this analysis, I first determined what the component parts of each game were. I then determined whether a particular component was functional or expressive. I then compared Boomshine's components to ChainRxn's components. For ease of reference, I have summarized this Section's component-by-component analysis in the table attached hereto as **Exhibit 13**.

32. This Section contains a number of screenshots of the Boomshine and ChainRxn games. Like all screenshots and figures in this declaration, I personally captured these screenshots. For the ease of the Court, I used Adobe Photoshop to illustrate particular game elements referred to in the text of the report. The figures in this section compare the functional screens between the two games addressed in this analysis. The game screens compared are the Title Screens, First Intro Screens, Game Screens, Level End Screens, Subsequent Level Intro Screens, and Game Completion Screens. In each Figure the screen elements highlighted by interconnected circles illustrate functional elements common to both games that are expressed differently in each. (Note that in each comparison Figure, the Boomshine screen is on the left and the ChainRxn screen is on the right.)

a. Title Screen Comparison.

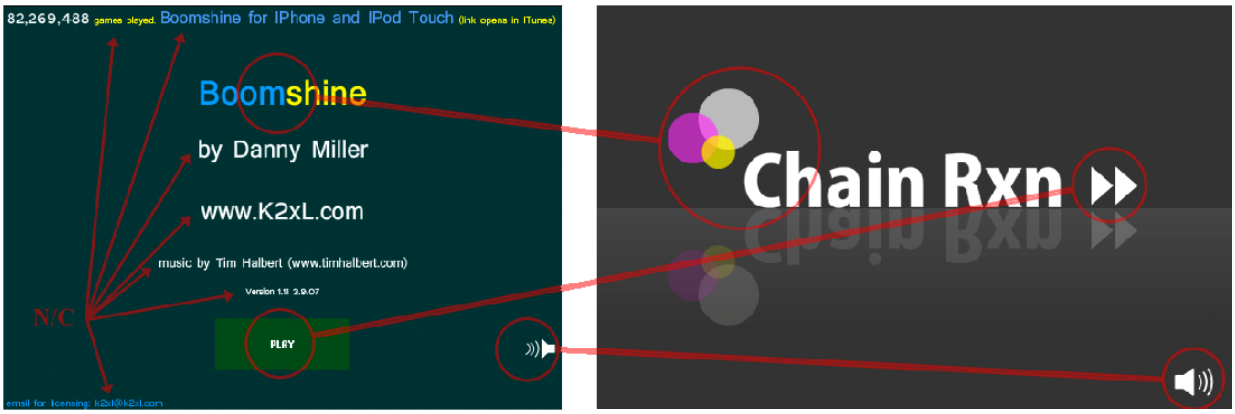


Figure 6: Title Screen Comparison

33. The ChainRxn game (right side of Figure 6) features a graphic three-ball logo and white game name styled to appear to be mirrored in a reflective surface. There is an animating play icon next to the game name which moves left and right to get the player's attention. There is a sound control icon. The Boomshine game (left side of Figure 6) expresses its title page as a predominately text-based screen without game logo or graphic. The game title is made from the colored letters "Boom" in blue followed by "shine" in yellow. The game "PLAY" button is a green rectangle with a tiny uppercase text caption. There is also a sound control icon, but one which is quite different from that seen in ChainRxn. The Boomshine Title Screen also includes a

great deal of text information which has no comparable (N/C) expression on the ChainRxn title screen. This text information includes the author's name and web address, an iPhone upsell link, a game play counter, a licensing link, music composer credit, and a version number.

34. The Title Screens in both games perform the same two functions: Starting the game and muting the sound effects. Despite the functional equivalence, from this side-by-side comparison ***it is clear that both games embody completely different expressions of Title Screen functions.***

b. First Intro Screen Comparison

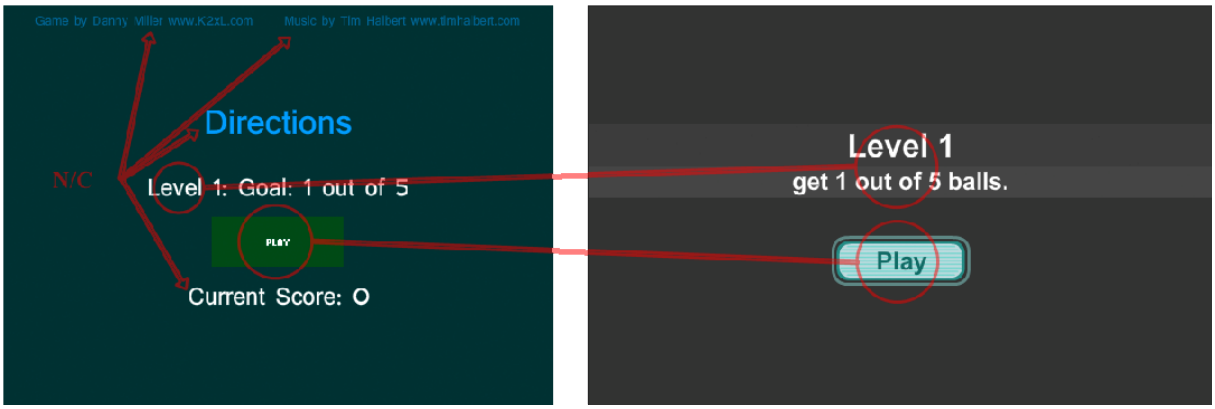


Figure 7: First Intro Screen Comparison

35. In a level-based game, the first screen following the Title Screen performs the function of setting out the game play goal of the upcoming level. It also functionally needs to provide a way to start playing the level. In most games the First Intro Screen contains a more verbose description than Subsequent Intro Screens since the player only needs to receive certain instructions once. The ChainRxn game (right side of Figure 7) has a very simple First Level Intro Screen which explains the level goal in a clear plain-language sentence. It implements the "level start" function through the use of an attractive rounded-rectangular "Play" button. For the Boomshine game (left side of Figure 7) the game designer chose a more abbreviated explanation of the level goal in a style similar to other on-screen text throughout the game. Here the "level start" function is implemented with a green rectangle with a tiny uppercase text caption ("PLAY") similar to that on the game's Title Screen. The Boomshine First Intro Screen also includes text information which has no comparable expression on the ChainRxn title screen. This

text information includes a repeat of the author's name and web address, a repeat of the music composer credit, and the current score.

36. The First Intro Screens in both games should be expected to have a similar approach, since they perform the same two functions: Stating the level goal and starting the level. And since both games embody the simplistic game play common to the Chain Reaction Genre, they are doubly likely to have some commonality. After all, there are very few ways that the game goal (click to try to explode n of m balls) can be stated. And despite the functional equivalence, and the limitation that the genre forces on the goal description, from this side-by-side comparison ***it is clear that the two games embody completely different expressions of the Intro Screen functions.***

c. Music Comparison

37. Background music is functionally optional for a game such as these, so it falls squarely in the area of expression. The designers of Boomshine chose to add background music to his game; the designer of ChainRxn chose not to do so. Since one game has music and one game does not, ***a clear difference in the musical expression exists between the two games.*** In many ways, the experience of the game differs substantially because the "mood" of the game is changed because of the use of sounds.

d. Sound Effects Comparison

38. Sound Effects are entirely an expressive game element in a game such as this. From a design standpoint, if a game designer chooses to include them each game activity (clicks, touches, collisions, explosions, etc.) should be accompanied by a sound effect. The sound effects help frame the feel of the game.

39. The designer of Boomshine chose a Sound Effect style compatible with the game's background piano music such that each collision invokes a sound using the tone and voice of an "electric piano". The combination of background music and compatible sound effect style creates an orchestral mix similar to two pianos playing together.

40. The ChainRxn sound palette is completely different, with no background music compatibility required. This game features a Sound Effect style that uses simple random

frequency pure-tone notes creating an effect that most closely resembles a wind chime and not a piano.

41. Sound Effects in games such as these are expressive, and each of these two games express the Sound Effects differently.

e. Game Screen Comparison

42. The Game Screens of both Boomshine and ChainRxn implement the same Chain Reaction game concept. Both provide a facility for choosing the location of the triggering event (a cursor). Both have a multitude of volatile objects moving about a single game playfield which will explode if touched by another explosion. And both have an on-screen score display to keep the player informed of his progress within the level. Those are all functional elements necessary to the game concept. As the following Figures demonstrate, ***each and every one of those functional elements are expressed differently in the two games.***

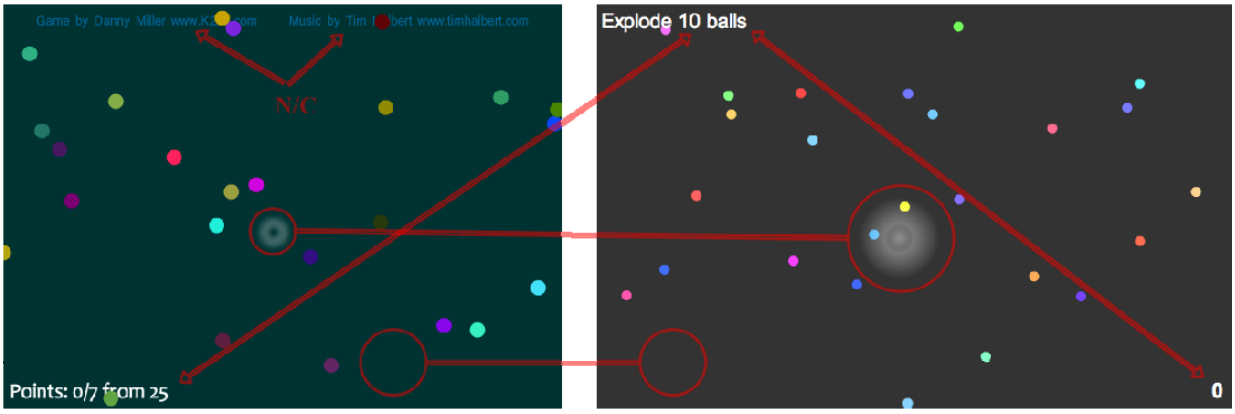


Figure 8: Game Screen Composition

43. The first difference to be noticed in Figure 8 is the different game background. It is common for a simple game to be played on a featureless colored background. Each game uses a different color for its game playfield: Boomshine uses a dark blue-green while ChanRxn uses a dark gray. These represent different expressions, not simply because the game designers each chose a different color, but also because colored game objects tend to have an increased "readability" on a screen with no color of its own (such as gray or black).

44. The next expressive difference is seen in the cursor shape and color. (There is also a dramatic size difference which is shown in Figure 9 and described in paragraph 46.) The

Boomshine cursor is small, dark blue-green in color, and is donut shaped with a hollow center. The cursor used by ChainRxn is quite large, medium gray in color, and is entirely filled in. The cursors in both games are rendered with a degree of transparency rather than opaque. This is a functional feature, required by the genre. The cursor exists to indicate where the first explosion will take place, and the player needs to be able to simultaneously see the moving cursor and watch the object patterns. If the cursor were to be opaque it would obscure part of the playing field and a number of objects, resulting in a different game play mechanic.

45. On-screen scoring is implemented differently as well. Language and wording differs, as does text positioning. ChainRxn displays the level's accumulated score on the bottom left of the page and lists the objective in the top right. These features are unique to the ChainRxn implementation. In contrast, Boomshine again repeats the author's name and web address, as well as the music composer credit – two elements not found on the ChainRxn screen. Boomshine lists the objective and the score together in an integrated way in the bottom right hand corner. Boomshine also uses the word "points" where ChainRxn does not.

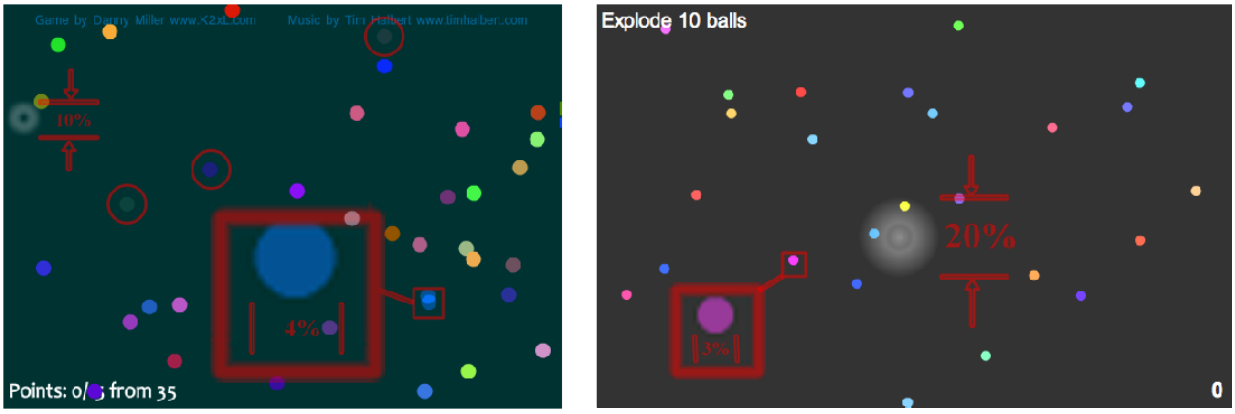


Figure 9: Game Screen Object Size and Detail

46. More Game Screen differences can be seen in Figure 9. The difference in the cursor color and graphic was noted above, and here we see the difference in size. At 20% of the screen height, the cursor in ChainRxn is fully twice the size of the cursor in Boomshine (10%). This difference can actually fundamentally affect the game mechanics of the game.

47. The game objects in ChainRxn are 25% smaller than those in Boomshine. This is expressive in two ways: visually and in game mechanics. A smaller object has a greater chance of passing by and just missing a decaying fireball, possibly requiring a higher level of skill in the game. The balls also look smaller than in Boomshine.

48. Figure 9 also shows that both game designers chose to represent the volatile objects in multiple colors. "Chaos Theory" illustrated that the color of the objects is not functional. Any round object will do. But while both of these games express the objects with color, they each do so with a different set of randomly generated colors.

49. Refer to the circled objects in Figure 9. Within the first few minutes of playing Boomshine the player is faced with objects that have so little color contrast against the game playfield that many objects are nearly rendered invisible. This effect is a combination of the color of the object and the luminance (brightness) component of that color. The designer of Boomshine appears to have selected a random color for each object from the widest range of available colors, with little thought as to the luminance component of that color. The result is a set of colors that include dark pastels, some of which are so close to the background color as to be difficult to see.

50. The objects in ChainRxn appear to use an entirely different set of colors. Throughout the game, all objects are limited to bright colors and form a high contrast against the game playfield. The objects in ChainRxn, as a result, are much easier to see for players of the game. One skilled in the art of video game design might say that more highly visible game objects are superior to hard-to-see objects. But even without passing judgment on the color set, each game clearly expresses object color in extremely different ways.

51. To summarize this section, the Game Screen represents most of the interaction between a game and its player. It is here that the fundamental game procedures occur and expression associated with game procedures manifest themselves. And this declaration has illustrated that ***there isn't a single element on the Game Screen that is not expressed differently by each of the games Boomshine and ChainRxn.***

52. As a side note, many of the functional elements described to this point are not just functional in the context of this particular game genre/idea, but they are also functional in the

sense that they are necessary components in any video game. A player wants to know how well he is doing in a game, therefore the game needs to have some kind of scoring or progress system. A player needs timely feedback as to the benefit of various actions, so his score is presented to him via an on-screen score display. Functions like these have been elemental to video games since the earliest single screen games. Accordingly, a game would only be notable if it *didn't* have a Title Screen; a level-based game would only be notable if it *didn't* have a Level Into Screen; it would only be notable if it *didn't* declare the score for a particular level; it would only be notable if it *wasn't* played on a rectangular screen with an on-screen score display, etc. Such features have been a part of video games for as long as there have been video games, and they are functional elements regardless of the genre of the game in which they appear.

f. Objection Motion Comparison

53. Functional to these games' shared game concept is the idea that the balls begin a level with random position and trajectory, and when their motion takes them to the edge of the game screen, they reflect back into the game playfield. The primary skill component of the game concept is predicting patterns and future convergence of the objects. Size and speed of the objects affect the difficulty and skill level required to play, and are therefore expressive elements.

54. The relative sizes of game objects is shown in Figure 9 and discussed in Paragraph 47. Let's move on to object motion. The designers of each of these two games chose decidedly different object size and speed to create the desired result in their games. Specifically, the objects in Boomshine move at an empirically-determined 42 pixels per second, while those in ChainRxn appear to travel at 56 pixels per second (an increase of 25%). As a result of the game designer's choice of object size and speed, the objects in Boomshine are 25% larger and 25% slower; the objects in ChainRxn are 25% smaller and 25% faster. Elements such as object speed and size (which effect collision radius) are the type of elements reserved by game designers to "tweak" game mechanics. Obviously smaller / faster objects interact differently from larger / slower objects. It is through the tweaking of these game play elements that a game designer will hone his implementation of a set of game procedures.

55. Also related to object motion is the notion of the objects bouncing off the walls into the zone of game play. At the risk of repeating myself, if the balls did not bounce off the game's "walls" the result would be a completely different game concept and game mechanic. By reflecting back into play when they reach the edge of the screen the objects set up the patterns that are at the heart of the game play and learnable skill. This rebounding action limits the play to a single screen, limits the play to a fixed set of objects, and creates a pre-defined and deterministic future pattern of the objects.

56. A close inspection of this "rebounding" effect shows that even this fundamental aspect of the game concept leaves room for individual choices to affect game play. Those expressive differences show up in Boomshine and ChainRxn through the differing game playfield dimensions, and the software limits in the game's implementation of the reflection of the object.

57. It might be helpful for the reader to understand how a reflected object's motion is deterministic. A moving object on a computer display that bounces off all four screen edges will follow a known and predictable path. Regardless of its starting angle or speed, its position on the screen can be computed in advance. Furthermore, its path represents a finite loop. There will be a point in the future where the object will end up positioned at the exact pixel from which it started, and begin to retrace its initial path. The mathematical formula to describe the object's path would relate the object's angle of travel to the least common denominator of the ratio of playfield width and height. In a non-mathematical sense, the path inscribed by a moving object will be dramatically affected by the size and shape of the playfield, as well as potentially the size, shape and speed of the moving object.

58. The actual mathematical formula to describe the object's path and its loop point is beyond the scope of this document. But a real-world example might assist in the visualization of the effect. Shoot a billiard ball at a specific angle on a standard six-foot pool table and it will bounce off certain cushions. Shoot that same ball at the same angle on a nine-foot Snooker table and the result will be dramatically different. In some cases it will hit a side wall first on one table and an end wall first on the other. Take away friction, cushion compression, spin, and loss of momentum (as is done with perfectly elastic bounces in video games) and the ball will continue

to bounce around until it returns to its starting point. But through its travels the pattern formed by the same object on the two different playfields are dramatically different.

59. This pattern difference shows up in the two games central to this analysis. Boomshine's visual game playfield is 550 pixels wide X 400 pixels tall. The visual game playfield of ChainRxn is 626 pixels wide X 400 pixels tall. The almost 14% increased width of ChainRxn's playfield makes for a substantial difference in the patterns formed by its moving objects. By itself this difference in width changes the pattern as in the pool table analogy above. But a closer inspection reveals that the playfield size difference is exacerbated by a difference in the rebound code as implemented by both games:

60. The objects in Boomshine reflect off an imaginary boundary that does not precisely align with the playfield's visual edges. It appears that the software decision to reflect is made when the center of the object reaches the edge of the playfield. This results in a portion of the object image leaving the screen bounds before it is reflected.

61. The programmer of ChainRxn correctly took the radius of each object into account and implemented the code such that the rebound occurs when the *edge* of the object reaches the edge of the playfield, not the *center*. This results in the object appearing to bounce exactly off the screen boundary as would be expected in the real world.

62. This programming difference expands the Boomshine logical playfield by twice the ball radius in both width and height, adding 16 pixels to each dimension. Accounting for this difference, Boomshine's objects rebound around a logical playfield size of 566 X 416, causing an even greater difference in object patterns. Programming the objects to rebound after passing outside of the physical playfield represents another expressive element difference between the two games. A game programmer might even consider it to be a programming bug in the Boomshine code. In either case it is yet another factor affecting the object patterning in each game.

63. Here again, ***there isn't a single aspect of Object Motion that is not expressed differently by each of the games Boomshine and ChainRxn. Indeed, some of the elemental game mechanics are different because of the subtle design choices made by each developer***.

g. Comparison of the Number of Objects Per Level.

64. In a level oriented game, the functional reason for levels to exist is to provide an escalating level of difficulty as the game progresses. Missile Command, the first Chain Reaction Game, itself had escalating levels of difficulty, with each level featuring increasing numbers of projectiles. Here, both games implement this functionality by increasing the number of exploded objects required to advance to the next level of difficulty.

65. For a player to be able to explode a larger number of objects, there have to be a larger number of objects on the screen in each successive level. Of that pool of objects, however, the number of object explosions required to advance to the next level differs between the games. The numbers required by each game are reflected in attached **Exhibit 13**.

66. The difficulty of each level escalates – as can be said about virtually every level-oriented game ever made. The rate of the escalation of difficulty is expressed by the game designer when he sets level goals, and, as shown in **Exhibit 13**, ***the level goals set for these two games are significantly different.***

h. Comparison of the Scoring Systems.

67. In a score-based game, scoring is a functional element. Most games include scoring to help the player hone his skill and to differentiate between player results. Level-oriented games require the completion of a set goal before advancing to the next level, further requiring the player to replay the level if the goal is not reached. A level-oriented game keeps the level score as a separate value, only adding it into the total score if the goal is reached. All of these functions are embodied in these games. The actual mechanism for computing the player's score is an expression of the game designer.

68. Boomshine computes a level score using one point per object exploded.

69. ChainRxn expresses the game score with a calculation that includes time. Here the level score is computed based on the amount of time elapsed between the first player-induced explosion and the explosion of the object to be scored. This provides a time dimension to the game play. A player is not only rewarded for the objects which explode, but if the chain reaction

is sustained for longer periods of time, the player's score can increase exponentially. Boomshine has no such scoring mechanism.

70. Accumulating a conditional level score and awarding it to the player upon completion of a task is a fundamental function of level-based games. How that score is calculated is clearly expressive. ***Both of these games implement unique and different scoring algorithm.***

i. Level End Determination Comparison.

71. Both games are level-oriented and test during play whether the level goal has been reached. In a game of this genre it wouldn't be appropriate to stop the chain reaction when the goal is achieved, rather the chain reaction is allowed to continue and potentially rack up additional points. Instead the player is alerted that the goal has been reached (if only to terminate the stress of waiting), and the explosions continue.

72. By design, Boomshine defines the end of a level as the instant the last explosion *begins to decay*. At that instant all object motion stops and object collisions are disabled. At that same instant the screen begins a fade-to-black before launching the Level End screen. If the level is ending with the goal met, this transition is accompanied by an electric piano arpeggio. During the game play, at the time the goal is met the background color is faded from a dark blue-green to a lighter blue-green. At the same time the on-screen display indicates an explosion to ball ratio larger than one (i.e. 12/5).

73. By design, ChainRxn extends a level until after the last explosion has completely *decayed to nothing* (not as the last one begins to decay, as done in Boomshine). Objects continue to move and interact until the final explosion has disappeared. There is no fade-to-black, but instead an immediate transition to the "Level End" screen. No sound effect accompanies this transition, (a Level End musical stinger is played later during the Level End screen). While the level is active, at the time the goal is met the background color is faded from dark gray to light gray. At that same time the on-screen display (which has been saying "3 more balls…", "2 more balls…", "Last ball!") disappears.

74. In my experience as a programmer, I am aware of a general programming concept referred to as "O.B.O.E" (pronounced like the musical instrument). O.B.O.E. stands for "Off By

One Error". This error crops up because computer systems sometimes count elements beginning with the number "1", and sometimes beginning with the number "0". There is a subtle form of O.B.O.E in the Boomshine game.

75. The difference between Boomshine terminating a level at the *start* of the collapse of the last explosion, and ChainRxn terminating a level at the *end* of the collapse of the last explosion, is a subtle one. A non-programmer might not even notice the difference. But this difference opens a window into the program execution of both games. Whether this is a bug in the Boomshine code, or a conscious choice by its programmer, the fact that these games handle this transition differently speaks volumes to an experienced programmer. This kind of O.B.O.E. is very subtle and would get carried along whenever a section of code is copied and reused. That the Level End sequencing differs between the games is ***strong indication that the code, and the underlying game, were not copied.***

76. Both games use a background color change to represent the meeting of the level goal. This is a very simple programming technique used by many games throughout history (See **Exhibit 14**). In Flash this is done with a single line of code. Despite the fact that both games use a background color change as a game indicator, ***the overall expression of the Level End sequence differs noticeably between the games.***

j. Level End Screen Comparison.

Figure 10: Level End Failure Screens.

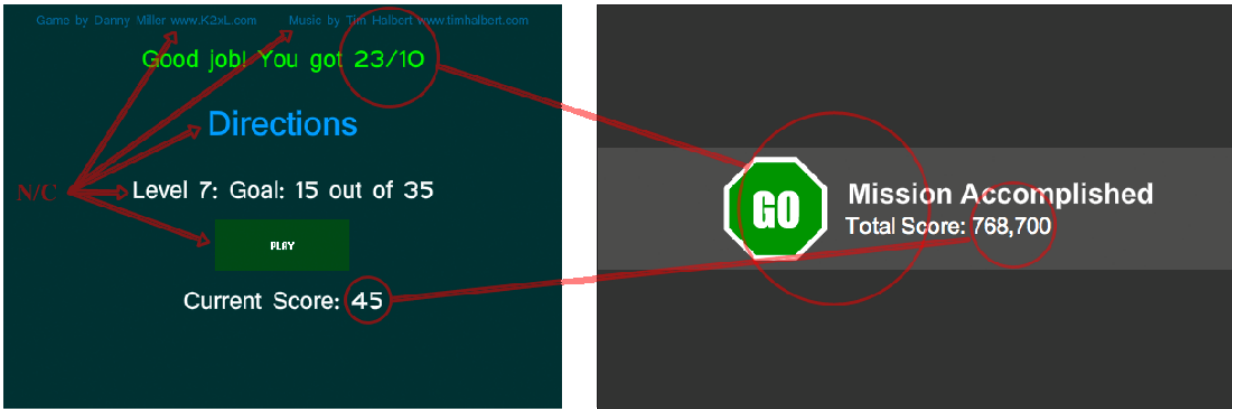


Figure 11: Level End Success Screens

77. Functionally a "Level End" screen wraps up the story just completed and typically summarizes the scoring or accomplishments. Here it is difficult to compare the games side-by-side because the Level End paradigm is completely different between the two games. The designer of ChainRxn chose to implement a unique Level End screen complete with international icon and "Good Sound / Bad Sound", followed by a separate screen to introduce the Next Level. The designer of Boomshine combined the Level End message with the Next Level screen with no sound effects. There are numerous differences in expression between the screens in Figure 10 and Figure 11.

78. ***The Level End screens differ not only in visual expression, but in logical sequencing as well***.

k. Game End Screen Comparison.

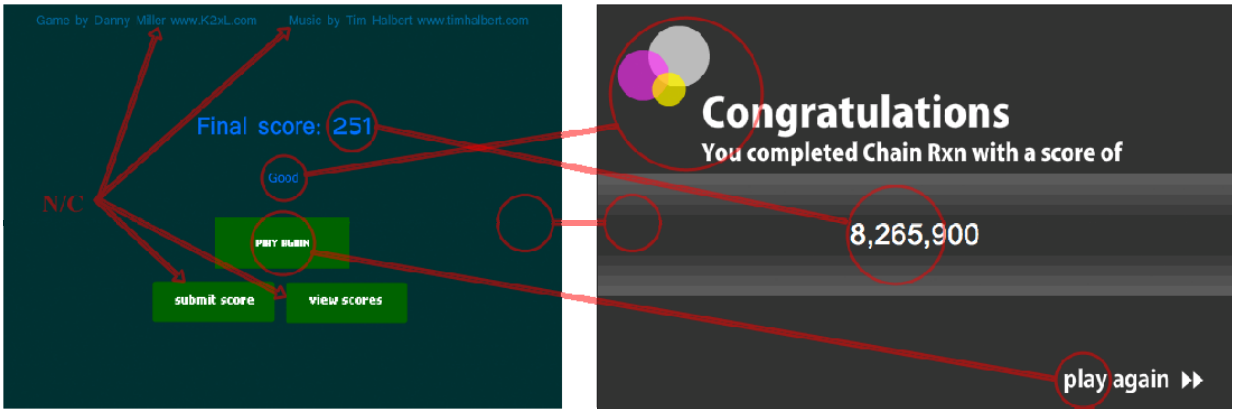


Figure 12: Game End Screen Comparison

79. Functionally, the Game Complete screen wraps up the experience. It should contain the final score, and it often provides an achievement level consistent with the story line (i.e. "You have achieved the rank of Captain"). Functionally it also provides a way to start the game over, i.e. "PLAY AGAIN".

80. Each game expresses the Game Complete screen functions very differently. The first obvious difference is in the background. ChainRxn introduces a gradient detail that separates itself from the plain background of Boomshine. ChainRxn reintroduces the 3-ball graphic logo as part of a congratulatory message. ChainRxn also displays the final score with white text prominently in the center of the gradient element, and places an animating "play again" button near the bottom-right corner of the screen.

81. Boomshine displays the final score in blue text following the caption "Final score". There is also a ranking word ("Good") which is changed based on a player's performance (a feature not seen on the ChainRxn screen). The author's name and web address, and the music composer credit are repeated on this screen, and there are two buttons added for High Score submission that have no counterpart on the ChainRxn screen.

82. Note also that the Boomshine screen has a PLAY AGAIN button similar in style to its earlier PLAY buttons. But this button contains a typographical error. It appears to spell the word PLAY as PlAY, apparently using a lowercase "L" within an uppercase word. This is particularly interesting where allegations of copying are made. ***If game code or screen was copied, one would expect to find the same bugs and typos in the version alleged to be copied.*** Later in this declaration is a more detailed analysis of the source code of both games that explores those allegations.

C. SOURCE CODE ANALYSIS

1. Boomshine and ChainRxn Use Flash Programming Code.

83. I have performed analysis and comparisons to determine whether any evidence exists that ChainRxn copied Boomshine's source code. The copying of the type of source code at issue would be the most significant indicator of copying.

84. Both of the games central to this analysis were written with the Adobe Flash Authoring Tool (aka "Flash"). The programming language built into Flash for games is Actionscript. The "open platform" culture that existed at Macromedia during the creation of Flash left the executable files somewhat open to viewing. It is possible to decompile the game code from the published executable, enabling access to the Actionscript programming code created by the game's author. Many people freely share Flash code, and in fact, there are thousands of lines of Flash code on the Internet to help fledgling programmers learn how to make games. It should be noted that decompiling Flash code is not a simple process available to the casual game player, but a highly technical process requiring special tools.

2. Brief Description of Code Obfuscation.

85. Code obfuscation is the process of transforming code into a form that is unintelligible to human readers while preserving the functionality and structure for computers.[13] The code is changed to hide its meaning, text strings are changed to a non-ASCII encoding, etc. Every trick pulled by the obfuscator makes the code harder to read by a human, but still operates correctly when run through the Flash player. Because the goal is to keep someone from knowing how the code works, typical obfuscation programs will turn the decompiled code into gibberish.

86. I have reviewed Mr. Miller's deposition transcript. According to that transcript, Mr. Miller testified in deposition that he passed his code through an obfuscation program prior to placing it on a website to be played. He also testified that he obfuscated the code before submitting it to the Copyright office as well.

87. Many people are concerned about the commercial value of their programs, and they don't share the desire to expose their programming techniques to the world. For those people, code obfuscators were created.

[13] This definition is taken from the webpage located at: http://en.wikipedia.org/wiki/ActionScript_code_protection. A true and correct screenshot of this webpage is attached hereto as **Exhibit 15**. This screenshot accurately portrays the webpage as it appeared to me through my web browser.

88. In this instance, the Boomshine code appears to be intact and workable, with only certain variables missing. If it is obfuscated code, it is an extremely poor quality obfuscation program.

3. Automated Source Code Comparison.

89. If Mr. Miller obfuscated his code before making it public, presumably his allegation of code copying refers to the public code. Therefore a decompiled version of his publicly available SWF file was used as the basis for the source code analysis. (This decompiled code will be referred to as the "Public Boomshine Source").

90. Defendant Yeo's game code is not available to us. But an examination of his code from the Internet shows that his code has not been obfuscated. A decompiled version of the game code found at http://chainrxn.zwigglers.com/swf/chainrxn.swf provided a clean source code file for comparison. (This decompiled code will be referred to as the "ChainRxn Source").

91. The Public Boomshine Source and ChainRxn Source were passed through a commercially-available tool designed to identify piracy and plagiarism. The tool used is called CodeMatch and is part of a suite of tools called CodeSuite®.[14] "CodeMatch uses several algorithms to determine similarity between two source code files … When multiple files are compared, file pairs are given a correlation score between 0 and 100."[15] A score of 0 indicates that there are no matches of any part of any line in the file (a result that would be highly unlikely given two files written in the same human language); a score of 100 indicates an exact match.

92. 22,811 lines of code in 164 files were scanned for matches. The average score was below 10. Note that both games are written in Flash using Actionscript, so the language dictionary is the same. Two programs written in the same language will certainly show some similarities. But if blocks of code had been copied by one programmer from another, there would be large groups of files scoring near 100. This result is fairly convincing, indicating no code copying took place.

4. Manual Source Code Comparison.

[14] http://www.safe-corp.biz/products_codesuite.htm
[15] http://www.safe-corp.biz/CodeMatch_algorithms.htm

93. Rather than rely completely on a software tool to compare the code, I also personally looked at the source code and compared similar functions. For this I used the non-obfuscated code provided by Mr. Miller and that I understand to have been deposited with the Copyright Office as part of his copyright registration application for the Boomshine computer file. Taking this step addresses any possible copying of the Miller non-obfuscated code. It also gives me an opportunity to judge the code from a human perspective, applying a level of professional judgment not possible with a computer program's heuristics.

94. I found that there were even fewer actual similarities in the code than the software suggested. CodeMatch errs on the side of trying to find matches, and with a programmer's eye one can see that occasionally one or two similar instructions line up while performing entirely different functions. This manual scan also shows how each programmer structured his code, and these structures show no similarities not forced on the programmer by the constraints of Flash.

5. Example of Source Code Comparison Results.

95. Attached hereto as **Exhibit 16** is an example of the actual code used by each game to create a new game object. Referring to the left column of the table, the programmer of Boomshine refers to an object as a "cell". Focusing on a single aspect of this code, the color of a newly initialized cell is derived as a random number from 0 to 16777216 which allows for all 24 bits of color. In fact, from a close inspection of the highlighted code it appears that an object could end up with the exact color of the background and therefore be completely invisible.

96. Referring to the right column of **Exhibit 16**, the programmer of ChainRxn named his function "newball". Looking at the way this code implements the new ball color (highlighted), a HSV (Hue Saturation Value) color encoding is used. A random Hue and Saturation are chosen while keeping the Value (or Brightness) set to maximum. That HSV value is converted to an RGB value for the object, but using this method guarantees a completely different set of colors from Boomshine and keeps the contrast of the objects at a high level making the moving objects easier to follow visually.

97. Comparing these two code modules, there is no similarity in structure, philosophy, or execution between the method each game uses create a game object. There is clearly no

evidence of copying here. This is the result of two programmers developing an independent idea and implementing that idea based on their independent programming training and skill.[16]

6. Bug and Typo Carryover.

98. It is a well known doctrine of software piracy that when the same bug appears in two pieces of nearly identical software programs, it points to a likelihood of copying. I have identified three such bugs in Boomshine. In Paragraph 62 I identified a rebound error in the Boomshine code; ChainRxn has no such bug. In Paragraph 82 I identified a typographical error in the word "PLAY"; ChainRxn has no such bug. In Paragraph 49 I explain about a color error that I noticed while playing the game, and the programming code described in Paragraph 95 supports my observation. ChainRxn does not share this problem.

99. There are no instances where a bug or typo in Boomshine can be found in ChainRxn. This is more compelling evidence that there was no copying of code between these games.

7. Summary of Source Code Analysis.

100. To address the allegation of code copying I brought to bear three powerful tools: A high-end commercial plagiarism detection tool, a black-box analysis looking externally at bug and typo carryover, and my own expertise from 40 years of programming (34 of which as a video game programmer and designer).

101. ***There is no evidence of any source code copying between the two games***.[17]

D. Parallel Development of Similar Games

102. There have been cases throughout the history of video games where two very similar game concepts were developed independently of one another. I was involved in at least two such examples – here is an anecdotal description of one of them. The time was 1981, when video game design took place in secret laboratories and we were not allowed to discuss a game idea even among our friends. Game development in the 1980s could take a year or more in total

[16] The different ways in which this code is written in the Boomshine and ChainRxn games also demonstrates that the visual representation of the colored objects in each of those games will be different. For instance, ChainRxn's code will (and does) result in smaller, brighter balls with differing hues and color saturation levels than those implemented in Boomshine.

[17] All source code decompilation files and output are hereby attached as **Exhibit 17.**

secrecy, so any games appearing within a year of each other were clearly developed independently.

103. I developed a game called "Freeway", based on watching a man trying to cross 10 lanes of traffic near the Chicago Convention Center during CES. My game had a chicken trying to cross a Freeway crowded by cars. At the same time, 7,500 miles to the West in Japan an arcade game "Frogger" was being developed by Sega (to be licensed to Konami). "Frogger" arrived in the arcades around the time "Freeway" hit the shelves for the Atari home system.

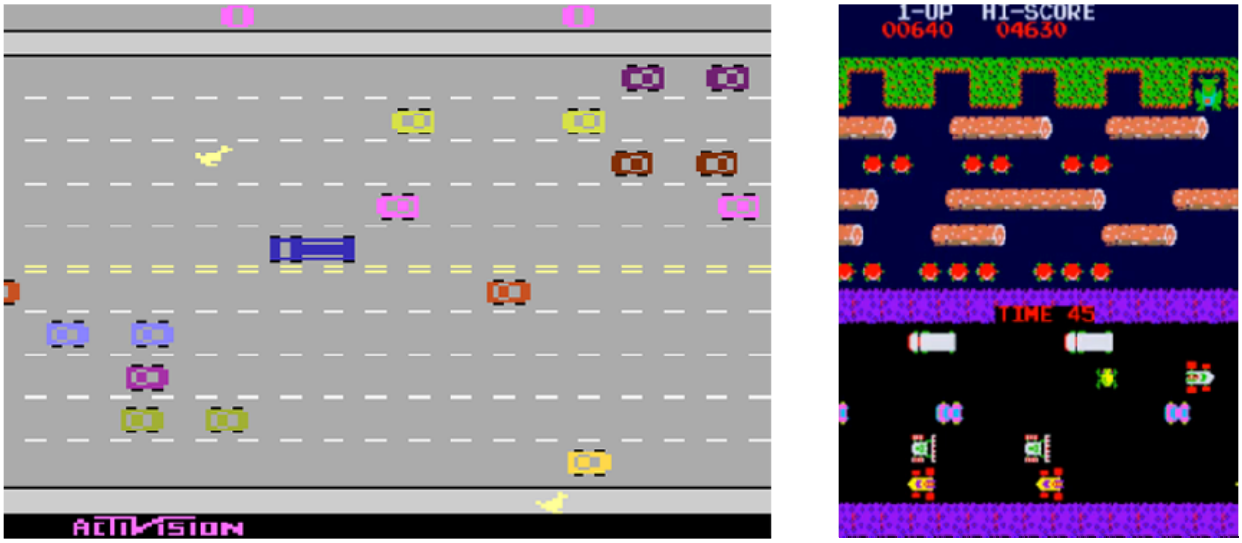


Figure 13: Freeway (left) and Frogger (right) developed independently

104. Developed independently over 7,000 miles apart, as seen in Figure 13, both games had the same basic game concept. The player tried to get his on-screen character from the bottom of the screen to the top, through traffic and other obstacles, as quickly as possible. Sequences of different length cars were on numerous lanes on the freeway in both games. The cars have multiple colors in both games. Both games are scored. They each have starting and ending sequences. Despite the remarkable similarities, the two games were developed entirely independently.

105. Interestingly, the main character in my game was a man until 3 days before code release. At that time the company's CEO and head of marketing suggested that we tie in the old joke "Why did the chicken cross the road?" I suspect that tired schoolboy joke was an American

cultural phenomenon unknown in Japan at the time. If it was otherwise, it is possible that there would have been two "Chicken crossing the road" games developed simultaneously.

106. How could two such similar games be developed simultaneously? They were both inspired by the same historical game: Space Race. After I decided to make a game about that guy running across Lake Shore Drive, I had to figure out how to represent the concept on a video screen. I drew upon my knowledge of how such a representation had been done in the past.

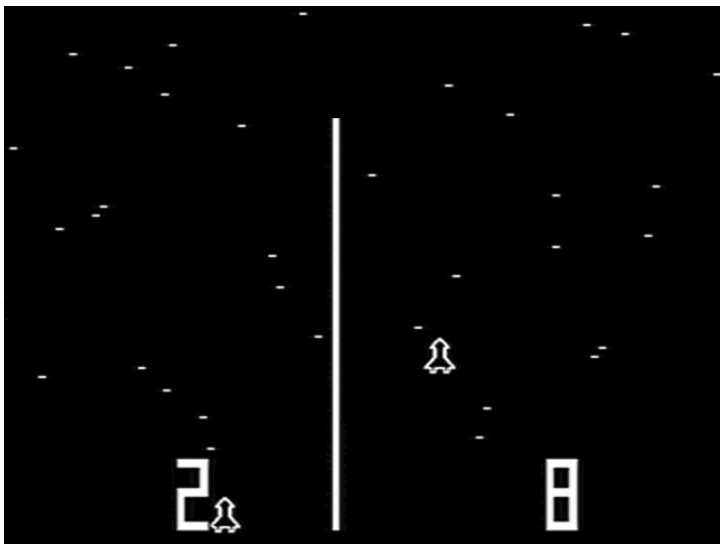


Figure 14: Space Race, a very early arcade game from the 1970s

107. In Space Race, as shown in Figure 14, players attempted to navigate their rocket ship vertically through horizontally moving "asteroids". Reaching the top of the screen earned a point, which was shown on the on-screen display. My expression of this game idea was different than that in Space Race. But the game concept was the same.

108. At the time that these two similar games were developed independently, both game designers could draw upon a few dozen video games in the public domain. At the time Boomshine and ChainRxn were developed, each game's designer could draw upon tens of thousands of games in the public domain which all shared similarity of concepts.

109. We have to consider the possibility that each of the two games Boomshine and ChainRxn could have been developed independently with no knowledge of the other. In fact, in

light of the facts that we have at this time, that possibility may be even more likely than Plaintiff's expert's "clone" argument.[18]

I declare under penalty of perjury under the laws of the United States that the foregoing is true and correct and that I executed this declaration on 3 day of March, 2011 at Menlo Park, California.

David P. Crane

[18] The gaming industry is replete with numerous examples of independently created, yet similar looking video games. For instance there are dozens, if not hundreds, of video games involving poker or solitaire or some other card game that all look virtually identical given the subject matter and the basic rules embodied in the visual elements of those games. Examples are endless.