KES Transactions on Innovation in Music: Vol 1 No 1
Special Edition - Innovation in Music 2013 : pp.79-93 : Paper im13bk-008

Game Scoring: *FEZ*, Video Game Music and Interactive Composition

Mack Enns

University of Western Ontario

1. Abstract

This paper is devoted in its entirety to the exploration of a new compositional mode called ‘game scoring.’ As with film scoring, game scoring supports, complements and elucidates the visual aspects of the gaming experience. This is, however, where the resemblances between game and film scoring end. There is a host of different technical and aesthetic obstacles and concerns that face the game scorer alone. Most significantly, innovations in game design have pushed game scorers to the point where much of their technique has evolved to resemble software programming more than any traditional compositional mode. To elucidate this point, this paper will use video game music composed by Rich Vreeland for the 2012 Xbox Live Arcade ‘puzzle-platformer’ game *FEZ*, as an instructive example.

2. Introduction

The video game industry is in the midst of transformation. Robert Abbott [1] writes that in its earliest years,

> ‘games usually had a top-down view that let you see the entire game board. The graphics were minimal. If the player was represented by a character on the board, that character was usually just a stick figure or a small round cartoon face. Most of these games relied on fast response, but the players could actually apply reasoning. Compared to today’s games, the 70s and 80s seem like a golden age of video games’ (para. 3)

Now, however, the video game has become a sophisticated art form in its own right, as well as the most profitable vector in the modern culture industry. In David O'Grady’s [2] words,

> ‘within the last decade, video games have emerged as a leading industrial and economic force, as a locus of cultural identity and exchange, as a site of great experimentation and proliferation, and as a subject of intense intellectual interest’ (para. 1)

Recently, much of the innovation in the video game industry has come from independent designers. Since they lack the massive marketing budgets of major gaming corporations like Electronic Arts and Microsoft, independent game developers have only innovation as a means of garnering attention in an increasingly crowded industry. Console games such as *Journey*, developed by Thatgamecompany for the PS3 provide not only non-linear narratives, but non-linear gameplay and a complete lack of textual instruction. Handheld games like *Super Stardust Delta*, developed by Housemarque for the PlayStation Vita utilize touchscreen and tilt controls to change perspectives, fire weapons and even manipulate time. These are just a few of the recent innovations in gaming that independent game developers are responsible for.

In fact, video game music has evolved alongside gameplay, though commentators tend to downplay or neglect the role music plays in the overall gaming experience.[1] In the last decade, musical composition for video games has become so technically and aesthetically distinct that I believe it requires its own category of technique. I call this category 'game scoring,' for lack of a better term. As with film scoring, game scoring supports, complements and elucidates the visual aspects of the gaming experience. This is, however, where the resemblances between game and film scoring end. There is a host of different technical and aesthetic obstacles and concerns that face the game scorer alone. Most significantly, innovations in game design have pushed game scorers to the point where much of their technique has evolved to resemble software programming more than any traditional compositional mode. I would go so far as to argue that in game scoring, distinctions between musical composition and software programming have blurred to a point where such distinctions are no longer of use.

This paper is devoted in its entirety to an exploration of this new compositional mode called 'game scoring'. As an instructive example, I use video game music composed by Rich Vreeland for the 2012 Xbox Live Arcade 'puzzle-platformer' game *FEZ*. This game received widespread acclaim upon release in 2012. The Imagine Games Network gave the game a rating of 9.5/10, and it currently holds a 90/100 on Metacritic. These are unheard of scores for an independent game. Eurogamer declared *FEZ* its Game of the Year, in fact, in December 2012, while the 2012 Independent Games Festival awarded it the Seamus McNally Grand Prize. Though the game is lauded, I chose *FEZ* solely for its utility as an instructional foil. This paper will begin with an introductory discussion of Vreeland's musical philosophy, in order to situate his

[1] This trend is especially apparent in reception-based scholarship on gaming from the realms of cultural, media and sociological studies. An assessment of these sources is beyond the scope of this paper, but would have to include: Jones [3]; Thornham [4]; Waggoner [5]; Wolf and Perron [6]; Crawford, Gosling and Light [7]; Domsch [8]; Egenfeldt-Nielsen, Smith and Tosca [9]; Ensslin [10]; Gamboa Villafranca [11]; Garrelts [12]; Huntemann and Aslinger [13]; Juul [14]; Martin [15]; Ruggill and McAllister [16]; Whalen and Taylor [17]; Paul [18]; and Newman [19, 20]; among others.

compositional process as an innovative departure from previous forms of video game music composition. Next it will introduce the story, gameplay mechanics and visuals of *FEZ* in order to reveal the unique game world which Vreeland had to provide music for. The final and main section of this paper will conduct an examination of the programming system for *FEZ*, simply named *Fezzer*, and its integrated music composition system. In doing so it will posit the existence of this new compositional mode, namely, game scoring.

3. Disasterpeace and *FEZ*

3.1. Proximity Over Order

In a September 2012 workshop presented by Pyramind Studios and Game Audio Network Guild entitled 'Philosophy of Music Design in Games', Rich Vreeland [21] notes that

> 'it was quite an interesting challenge because […] instead of thinking about order, like when things happen in the music, it was more about proximity, like which notes do I want to happen near other notes so that they sound pleasing. Which is kind of a weird thing to think about in music' (29:41)

Vreeland is better known by his musician name Disasterpeace, and in this quote he refers to a specific instance in the production of the soundtrack for the 2012 puzzle-platformer video game *FEZ*. The video game is notable for its exploration of space as a malleable concept, as it introduces players to its own spatial logic and presents space as manipulable. With the game's soundtrack Vreeland not only complements this exploration aesthetically with the combination of 'chiptune' sounds and studio effects like reverberation and delay, but also incorporates this exploration into its production and composition. His quote from the 2012 workshop signifies his emphasis on a spatial rather than temporal conception of music, and coincides with his discussion of 'Music Gameplay' and the '*FEZ* Music System'. Vreeland discusses the composition of a track entitled 'Synch', which plays during a level in which the main character Gomez must climb to higher altitudes with the help of blocks which appear and disappear. The appearance and disappearance of such blocks is dictated by the rhythm of the music, and each block is has its own sonic signature in the form of a bright low-resolution synth array. In other words, music gameplay simply refers to the connection of gameplay and music. The player must navigate a space that is composed partially of elements whose appearances and disappearances are dependent upon the soundtrack. A spatial conception of music goes hand-in-hand with many of the gameplay and story elements in *FEZ*, as the following discussion will show.

3.2. A New Dimension

FEZ is developed by the independent software company Polytron Corporation, which includes the game's creator and designer Phil Fish and its programmer Renaud Bédard. The latter are responsible for most of the development of the game; Fish determines the creative vision for the project while Bédard implements it into his programming. These roles were made distinct due to Fish's desire for complete creative control. It was only until after the game's visuals were designed and programmed that Vreeland was invited to compose and produce the game's celebrated soundtrack. Therefore his task was to musically explore a pre-conceived world with its own spatial rules, limitations, aesthetics and logic, and to provide a soundscape to represent, complement and sonically elucidate that world.

The world of *FEZ* is highly dynamic. Its 'levels' consist of non-Euclidean spaces known as 'Rooms'. At the outset of the game, Gomez, the game's 'protagonist', is a two-dimensional creature who lives in a two-dimensional world. Much like the protagonist in the classic 8- and 16-bit *Super Mario Bros.* series, Gomez has impressive jumping abilities, which serve as the main element of gameplay in a world composed of various types of platforms. Eventually, after a short 'tutorial' introduction, Gomez encounters a mysterious being known as the Hexahedron, who grants him a magical fez hat that allows him to perceive a third dimension, which rotates the gamer's perspective at will. As Gomez experiments with his new ability the Hexahedron unexpectedly fractures and explodes, causing the game to glitch, freeze and reboot, complete with BIOS screen. Gomez awakens in his room with his ability to perceive and manipulate a third dimension intact, and is charged with the task of recovering the scattered fragments of the Hexahedron before the world is torn apart.

Even after Gomez acquires the ability to perceive the third dimension, gameplay remains largely two-dimensional. Depth, or the Z-axis, is only visible to the player in the rotation of perspectives, and is not a factor in the actual obstacles and chasms which Gomez must traverse. The player must manipulate these perspectives to explore the world of *FEZ* and collect thirty-two cubes in the form of 'cube bits', 'whole cubes' or 'anti-cubes'. In so doing, Gomez performs actions that would normally be impossible in a truly three-dimensional world. For example, he may be on a platform in which a rotation of perspective moves it to the other side of the screen, even though he has not moved at all (Fig. 1, 2).

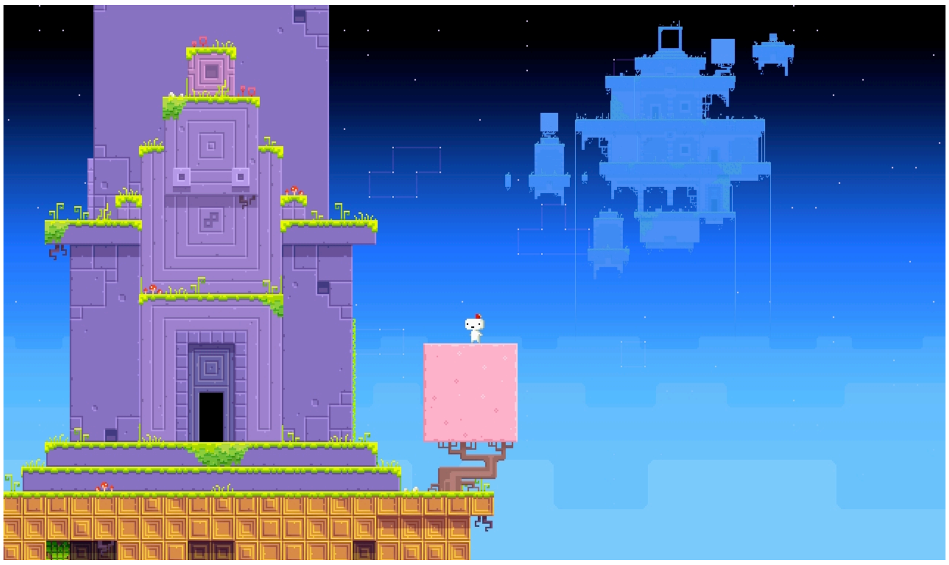


Fig. 1: Gomez atop a tree, unable to reach a higher ledge.

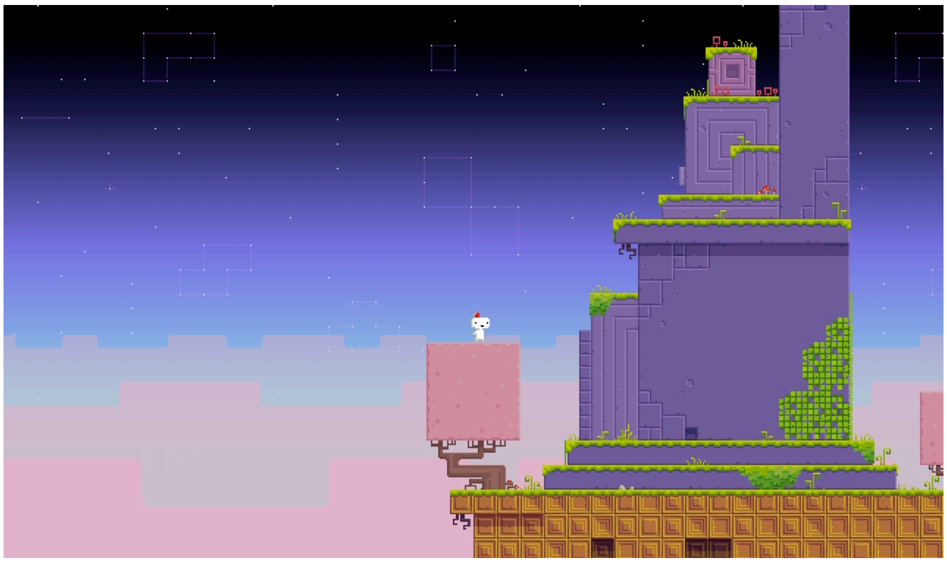


Fig. 2: Gomez atop the same tree as Fig. 1 with the perspective rotated. He is now able to ascend the tower.

Players of *FEZ* must conceive space in a different way in order to navigate its world. One of the first people who was allowed to explore this world—silently—was Rich Vreeland. Not only did Vreeland have to conceive of space differently in this initial run-through of *FEZ*, he was also forced to conceive of composition in a new way. The incorporation of the music system into *Fezzer*, the game's programming system developed by Renaud Bédard, allowed for this new compositional approach. Moreover, *Fezzer* was no less dynamic than the world it was used to create: as he composed, Vreeland was invited to propose ideas for the music system, which Bédard would then implement into his programming. The music system they eventually developed took the form of various tools and techniques integrated into *Fezzer* itself. Thus, the production and composition of the soundtrack was linked with the development of gameplay and design. The three tools and techniques from *Fezzer* which I will outline are (i) the sequence context menu; (ii) the scripts browser; and (iii) the main composition sequencer. These are names which I have attributed to these tools and techniques, rather than what Vreeland or Bédard may have called them.

4. *FEZ'* Music System Tools

4.1. Sequence Context Menu

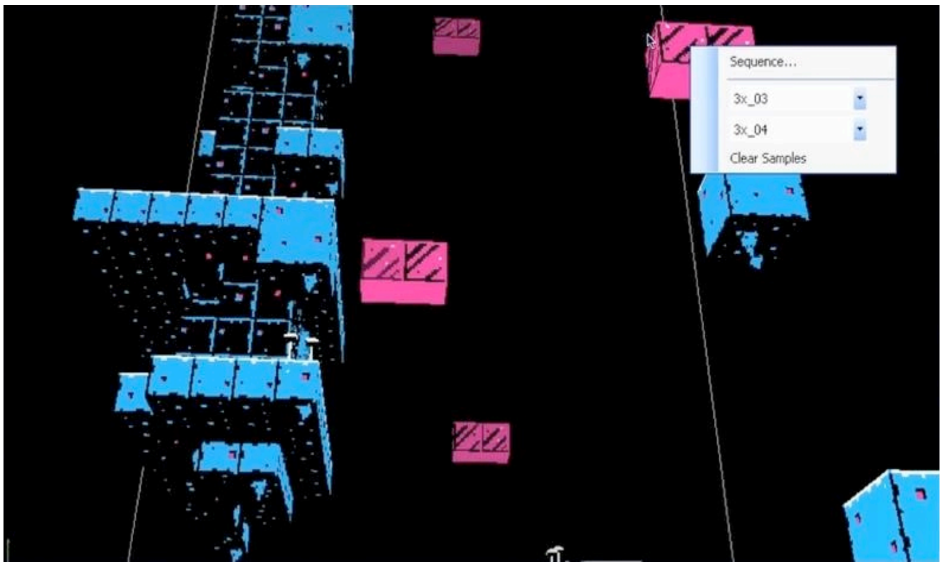
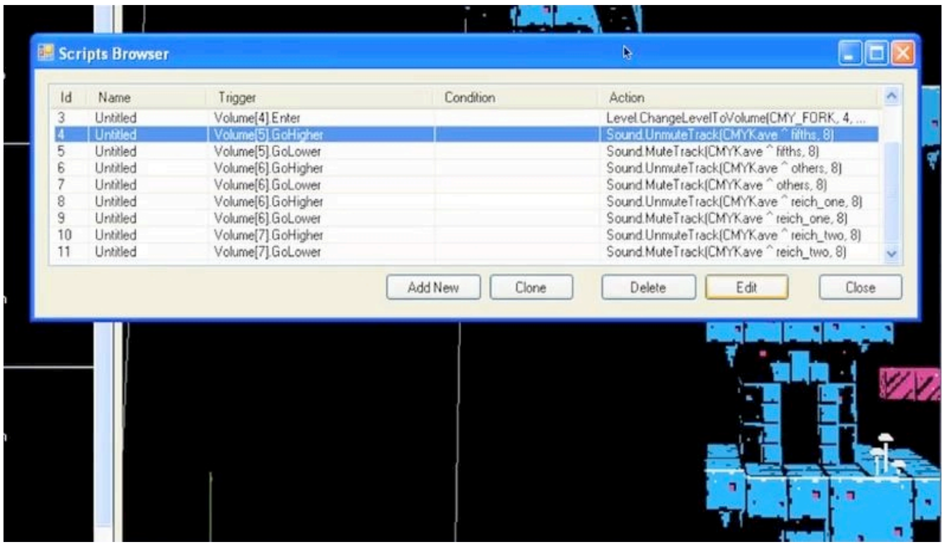


Fig. 3: The sequence context menu in *Fezzer*. One of the appearing and disappearing blocks in one of the Music Rooms is right-clicked, prompting the context menu.

Fezzer allows the user to physically explore every aspect of *FEZ* as an omniscient observer. Manipulation of perspectives is not necessary here as the user can already view any area in the full three dimensions. Right-clicking on any element in the game world, such as the block in Figure 3, prompts the sequence context menu. The sequence context menu is a tool specifically for the assignment of sounds to physical elements within the game. It is used for either sound effects or music given the scenario, or both as in the instance of Figure 3. The ‘Sequence…’ button allows the user to load a sample, piece of music or sound effect into the menu. In the above example, Vreeland has loaded ‘3x_03’ and ‘3x_04’ into the context menu as usable sound elements. These refer to the bright and bit-crushed synth arrays that coincide with the appearance and disappearance of bright red blocks in the Music Rooms.

This particular example involved considerable collaboration between Bédard and Vreeland. The former had to adapt the gameplay to the rhythm of the music composed by the latter. The blocks thus not only appear with the synth arrays, but appear on beat with the level’s score. Vreeland said that this process involved thinking about music in terms of proximity rather than order, or in terms of spatiality rather than temporality. The ability to visualize the implementation of music in *Fezzer* was indispensable in Vreeland’s composition process, as he could now think about ‘which notes do I want to happen near other notes so that they sound pleasing’ [21]. The word ‘near’ in Vreeland’s quote does not denote a nearness in time, but in (spatial) proximity between elements in the world of *FEZ*. Bédard’s music system allows for a spatial conception of music through its incorporation into the programming software itself. Right-clicking an element and assigning it a sound may seem like a simple task, but it also prompts a new way of thinking about music production and composition.

4.2. Scripts Browser



Id	Name	Trigger	Condition	Action
3	Untitled	Volume[4].Enter		Level.ChangeLevelToVolume(CMY_FORK, 4, ...
4	Untitled	Volume[5].GoHigher		Sound.UnmuteTrack(CMYKave ^ fifths, 8)
5	Untitled	Volume[5].GoLower		Sound.MuteTrack(CMYKave ^ fifths, 8)
6	Untitled	Volume[6].GoHigher		Sound.UnmuteTrack(CMYKave ^ others, 8)
7	Untitled	Volume[6].GoLower		Sound.MuteTrack(CMYKave ^ others, 8)
8	Untitled	Volume[6].GoHigher		Sound.UnmuteTrack(CMYKave ^ reich_one, 8)
9	Untitled	Volume[6].GoLower		Sound.MuteTrack(CMYKave ^ reich_one, 8)
10	Untitled	Volume[7].GoHigher		Sound.UnmuteTrack(CMYKave ^ reich_two, 8)
11	Untitled	Volume[7].GoLower		Sound.MuteTrack(CMYKave ^ reich_two, 8)

Fig. 4: The scripts browser window in *Fezzer*. This example is again from one of the Music Rooms, but the window may be pulled up in any area, as with the sequencer context menu.

Unlike the sequence context menu, the scripts browser window in *Fezzer* affects an entire room rather than just any one single element. Scripts are programs which are written for a specific run-time environment that can read and execute tasks in an automated fashion. In other words, scripts are sets of tasks that can be performed by programs that can interpret them, hence *Fezzer* deals its own specific type of scripts. The general nature of this definition points towards the wealth of possibilities with scripts, as they can perform almost any function so long as the host program can interpret them.

The scripts browser window in *Fezzer* is dominated by the presence of a table which lists each script's 'Id', 'Name', 'Trigger, 'Condition' and 'Action'. The 'Id' of a script is simply an identifying number, while the 'Name' column serves largely the same function. In Figure 4 it is safe to assume that Vreeland left the 'Id' and 'Name' fields at their default values. The 'Trigger' of a script is generally self-explanatory as that which sets the execution of a script in motion, but its implementation becomes more complicated in specific cases. Figure 4 shows a scripts browser window with scripts for one of the Music Rooms, which incorporates altitude-sensitive musical elements. As Gomez ascends higher in the Music Rooms in *FEZ*, as in Figure 5, different musical elements are added to and subtracted from the mix. Each trigger therefore indicates an altitude, signified by the 'Volume[x]' condition.

Furthermore, there is another condition which signifies whether Gomez is higher or lower than the specified altitude. For example, script four has 'Volume[5], GoHigher' as its trigger value, so any time Gomez goes higher than an altitude of '5', the script is triggered. The numbers which denote altitude are arbitrarily assigned to invisible blocks in the Music Room, which are positioned by Bédard. The 'Condition' column of the scripts browser allows for any other conditions to be entered, such as time of day or perhaps the amount of cube bits Gomez has acquired. In Figure 4 no extra conditions are necessary, so the column remains unused. Finally, the 'Action' column refers to what action will be taken when the trigger's conditions are met.

It may be helpful here to reiterate the flexibility of scripts, and note that they are governed by their own scripting language. 'Volume[x]' thus refers to altitude in the trigger field, rather than the volume of a sound, for example. The 'Action' column uses this same scripting language in the form of '[Target type].[Action][Target type 2]([Name], [Number of bars]', where 'Target type' is the type of element being acted upon, 'Action' is the action to be taken, 'Target type 2' is the sub-type of element being acted upon, 'Name' is the name of that element and 'Number of bars' is simply the length of the element. Script four, for instance, performs the unmute function on the loop 'CMYKave ^ fifths', which is 8 bars long.

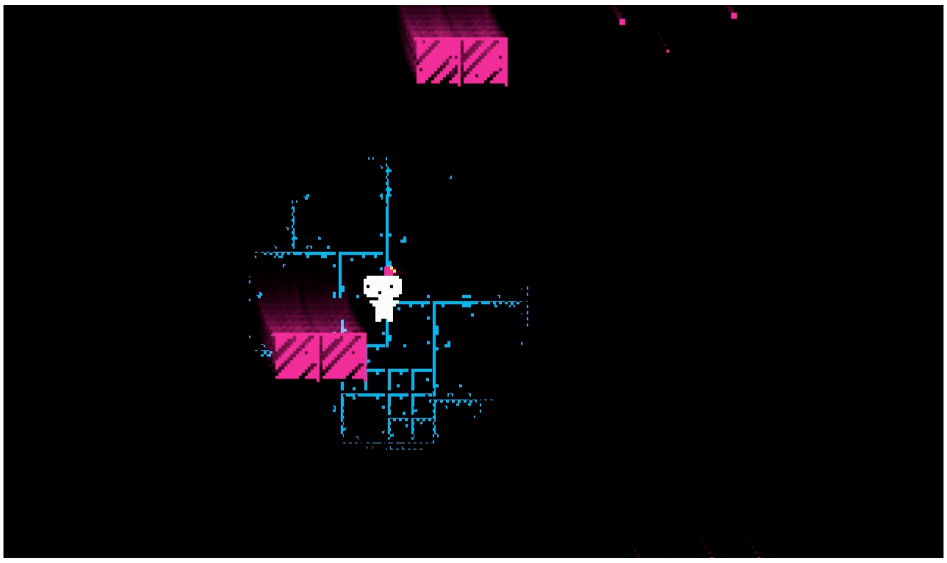
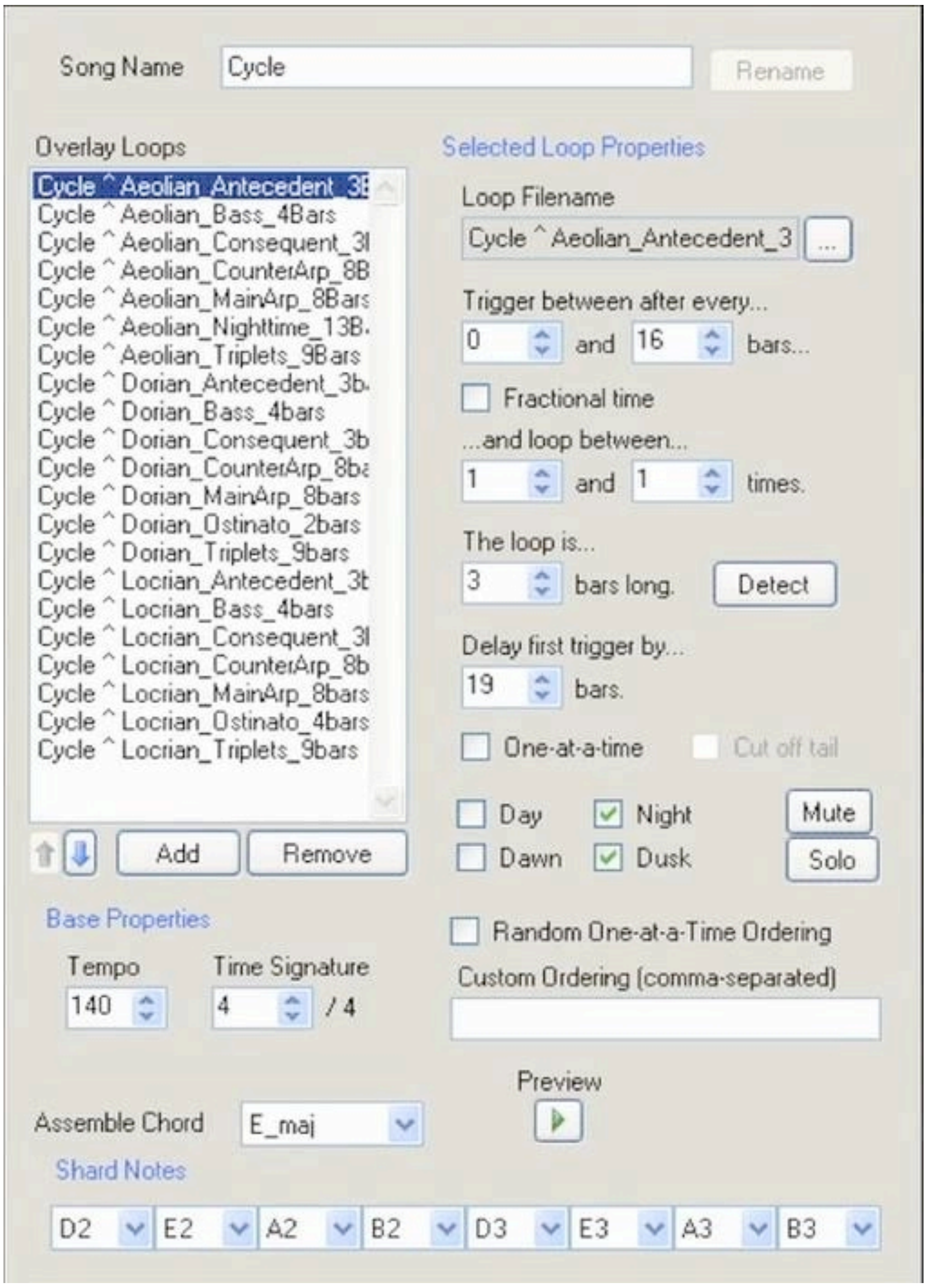


Fig. 5: Gomez ascends the first Music Room by jumping to bright red blocks as they appear.

If Gomez ascends higher than the altitude marked by an invisible block as ‘5’, a new musical element will therefore enter the mix, and it will remain there unless Gomez descends lower than the marked altitude. When this happens—that is, when Gomez descends below the designated altitude—gamers hear the opposite effect: the loop is muted again. In this sense, Vreeland’s ‘composition’ for the game is actually interactive, what Vreeland calls ‘Music Gameplay’. Progress in the Music Rooms is signified by the soundtrack, which rewards players with more elements of the song as they approach the summit.

I would like to provide a little more detail about how the 'gamer' influences the music here. This is, however, beyond the scope of this paper, and I intend to examine these topics in future work. For now, I am most interested in simply establishing the broad compositional parameters of ‘game scoring’; anything more would require time and space beyond what is appropriate here.

4.3. Main Composition Sequencer



Song Name: Cycle

Rename

Overlay Loops

Cycle ^ Aeolian_Antecedent_3E
Cycle ^ Aeolian_Bass_4Bars
Cycle ^ Aeolian_Consequent_3I
Cycle ^ Aeolian_CounterArp_8B
Cycle ^ Aeolian_MainArp_8Bars
Cycle ^ Aeolian_Nighttime_13B
Cycle ^ Aeolian_Triplets_9Bars
Cycle ^ Dorian_Antecedent_3b
Cycle ^ Dorian_Bass_4bars
Cycle ^ Dorian_Consequent_3b
Cycle ^ Dorian_CounterArp_8ba
Cycle ^ Dorian_MainArp_8bars
Cycle ^ Dorian_Ostinato_2bars
Cycle ^ Dorian_Triplets_9bars
Cycle ^ Locrian_Antecedent_3t
Cycle ^ Locrian_Bass_4bars
Cycle ^ Locrian_Consequent_3I
Cycle ^ Locrian_CounterArp_8b
Cycle ^ Locrian_MainArp_8bars
Cycle ^ Locrian_Ostinato_4bars
Cycle ^ Locrian_Triplets_9bars

Add | Remove

Selected Loop Properties

Loop Filename: Cycle ^ Aeolian_Antecedent_3

Trigger between after every... 0 and 16 bars...

Fractional time: ☐

...and loop between... 1 and 1 times.

The loop is... 3 bars long. Detect

Delay first trigger by... 19 bars.

One-at-a-time: ☐ Cut off tail: ☐

Day: ☐ Night: ✓
Dawn: ☐ Dusk: ✓

Mute | Solo

Random One-at-a-Time Ordering: ☐

Custom Ordering (comma-separated):

Base Properties

Tempo: 140 Time Signature: 4 / 4

Preview

Assemble Chord: E_maj

Shard Notes: D2, E2, A2, B2, D3, E3, A3, B3

Fig. 6: The main composition sequencer window. This example is from work on one of the Puzzle Rooms, which uses the song ‘Cycle’.

The main composition sequencer window is used mostly to determine the timing logic of the elements in one of Vreeland's 'songs'. Like the scripts browser, the main composition sequencer can make changes that affect an entire level, but also, like the sequence context menu, it can be used to tweak single musical elements. The song name can be entered or re-entered at the top of the window. The 'Overlay Loops' list box displays all the loops that can be in a level's song, which can be added, removed and reordered with the buttons at the bottom.

Although it is not readily evident in Figure 6, I assume that you may select and manipulate more than one loop at a time for faster workflow. Vreeland's naming style for his loops can be seen in the above example, and takes the form of '[Song Name] ^ [Mode]_[Musical Element]_[Amount of bars]bars'. All the information of a given loop is in the file name, so there is no guesswork necessary to determine which loop is which. It is notable that the 'Musical Element' field does not adhere to any specific type of musical aspect, but instead serves solely to help programmers identify the loop. In some cases, it identifies a type of instrument featured, as in 'Bass', while in other cases it identifies a melodic phrase in relation to others, such as 'antecedent' and 'consequent'.

The 'Selected Loop Properties' area serves most of the functionality of the main composition sequencer window. The 'Loop Filename' is visible at the top, with a browse button beside the text field. The 'Trigger between after every…' area has two text fields, with scroll arrow buttons, where a range of bars may be entered. In Figure 6 the song 'Cycle' is split into many overlay loops, which play in the Puzzle Rooms according to the settings entered here. The 'Trigger' section, for instance, denotes where the selected loop will play, within a given range if desired. This makes the actual song heard during gameplay slightly unpredictable, or aleatoric, as loops may come and go anywhere within these set ranges.

In fact, the listening experience in playing video games is largely different from the listening experience associated with records, which hold songs that are composed the same way every time they are played. With the music system in *Fezzer*, Vreeland can compose in a partially aleatoric style. The 'chance' aspect is, in large part, controlled by ranges programmed into the main composition sequencer, but also in part randomized because the loop can start any time within that range.

Below, the 'Fractional time' checkbox allows for irregular time signatures to be used in the deployment of loops. The '…and loop between…' section includes another pair of text fields with scroll arrow buttons. These can be set to a range of the amount of times the selected loop will play—another instance of aleatory composition. The length of the selected loop may be entered in the 'The loop is…' field, or it may be automatically supplied by the 'Detect' button. Vreeland's naming style incorporates the length of the loop in bars, so it is likely that he never uses the 'Detect' button. The 'Delay first trigger by…' field can be set to denote the number of bars after which the loop is played the first time. In this case, loops may be staggered in order to adhere more to the logic of a traditional song form.

The ‘One-at-a-time’ checkbox is oddly placed, as its setting applies to the entire song file, instead of just the selected loop. This setting works in conjunction with the ‘Custom Ordering’ text field below it, and allows the user to restrict the song to play only one loop at any given time, while the ‘Custom Ordering’ field dictates the order of those loops. Alternatively, ‘Random One-at-a-Time Ordering’ precludes the need for a custom order, as it plays loops one at a time at random. The ‘Mute’, ‘Solo’ and ‘Preview’ buttons are used to preview the song or selected loop within the main composition sequencer window. Finally, the time of day checkboxes ‘Day’, ‘Night’, ‘Dawn’ and ‘Dusk’ may be checked to specify when the selected loop may play according to the game’s time system.

The ‘Base Properties’ section of the composition sequencer allows for song-wide changes to be made to tempo and time signature. As with many settings in this window, these are musical elements which would normally be set in the compositional stages of writing music. In non-interactive sources of music such as records, the tempo and time signature are ordinarily set early in composition because they can dramatically change the form of the song. This follows a more traditional approach to composition because it is built upon the notion of a song’s ‘essence’ which is embodied in its notation or sheet music. In *FEZ*, the ‘essence’ of the song is in the gameplay because it is inextricably linked to it. Settings such as tempo and time signature must therefore remain malleable even late into the composition process. Alternatively, perhaps a better way to express this difference would be simply to say that the composition process must remain extended and ‘open’, right until the video game itself is complete.

The bottom section of the main composition sequencer actually deals with sound effects, as Vreeland wanted the eight cube bits that make up a full cube to have corresponding sounds that make up a full musical scale. The ‘Assemble Chord’ drop-down menu allows the user to choose the chord to be assembled, while each drop-down menu in the ‘Shard Notes’ area allows the user to choose a note for each cube bit to play.

5. Conclusion

A recent proliferation of independent designers and programmers has resulted in experimental and innovative tools and techniques for developing video games. Video game music has not been exempt from these innovations, but in fact has transformed and thrived under new methods of composition and production. Indie game development involves small work teams and a large amount of collaboration between designers, programmers and, most significantly for my purposes, composers. Video game music composition is now fully integrated into game programming, which makes the traditional distinction drawn between scoring and game programming all but irrelevant. I call the new ‘hybrid’ compositional mode ‘game scoring,’ and I would simply note that I think it differs markedly from any traditional modes of composition. The game scorer must traverse technical and aesthetic obstacles that depend on the medium’s interactive nature, which requires composers think as programmers rather than scorers. Indeed, innovation, at least for video game music of the current moment, requires integration. This integration has resulted in the phenomenon known as game scoring.

6. References

[1] R. Abbott. Video Games Are Incredibly Stupid! Logic Mazes. Web. 20 Nov. 2013. http://www.logicmazes.com/s7g2k/video.html

[2] D. O'Grady. 'Video Games: The State of the Field.' UCLA Game Lab. Web. 20 Nov. 2013. http://games.ucla.edu/resource/video-games-the-state-of-the-field/

[3] S. E. Jones. The Meaning of Video Games: Gaming and Textual Studies (2008)

[4] H. Thornham. Ethnographies of the Videogame: Gender, Narrative and Praxis (2011)

[5] Z. Waggoner. My Avatar, My Self: Identity in Video Role-Playing Games (2009)

[6] M. J. P. Wolf and B. Perron. The Video Game Theory Reader (2003)

[7] G. Crawford, V. K. Gosling and B. Light. Online Gaming in Context: The Social and Cultural Significance of Online Games (2011)

[8] S. Domsch. Storyplaying: Agency and Narrative in Video Games (2013)

[9] S. Egenfeldt-Nielsen, J. H. Smith and S. P. Tosca. Understanding Video Games: The Essential Introduction (2008)

[10] A. Ensslin. The Language of Gaming (2012)

[11] X. Gamboa Villafranca. Baroque Worlds of the 21st Century (2012)

[12] N. Garrelts (ed.). Digital Gameplay: Essays on the Nexus of Game and Gamer (2005)

[13] N. Huntemann and B. Aslinger. Gaming Globally: Production, Play, and Place (1st ed.) (2013)

[14] J. Juul. Half-Real: Video Games between Real Rules and Fictional Worlds (2005)

[15] J. M. Martin. Keeping Up with the Virtual Joneses (2012)

[16] J. E. Ruggill and K. S. McAllister. Gaming Matters: Art, Science, Magic, and the Computer Game Medium (2011)

[17] Z. Whalen and L. N. Taylor. Playing the Past: History and Nostalgia in

Video Games (2008)

[18] J. Newman. Best Before: Videogames, Supersession and Obsolescence (2012)

[19] C. A. Paul. Wordplay and the Discourse of Video Games: Analyzing Words, Design, and Play (2012)

[20] J. Newman. Playing with Videogames (2008)

[21] R. Vreeland. Philosophy of Music Design in Games. Web. 13 October 2013. YouTube. http://www.youtube.com/watch?v=Pl86ND_c5Og