All Together Now,

Using Multiple Frameworks to Inform Serious Game Design and Development

Declan A McClintock

Michigan State University

Author Note

The games discussed in this paper were used in an Institutional Review Board approved study at Michigan State University. The preliminary results of that study were discussed at the Foundations of Digital Games Conference 2021 (McClintock & Owen, 2021). However, a more complete and statistically sound analysis of the data collected from the study is still underway and has not been completed at the time of this paper's publication. The focus of this paper is to discuss the design and development process for the games as informed by existing frameworks. The author has no conflicts of interest to disclose.

Correspondence concerning this article should be addressed to Declan Andrew McClintock, Michigan State University. Email: mcclin61@msu.edu

Abstract

Existing serious game design interventions often focus entirely on the impactful results of the intervention, while leaving out the details of the design and development processes for the games used in the intervention. While the impact of the interventions is important in showing the power of serious games, leaving out a thorough discussion of the design and development process for the game(s) used in those interventions does a disservice to the field.

In order for future researchers to know what design and development practices to adopt to make successful serious games, game intervention studies need to include a detailed discussion of what frameworks informed their design and development. This will give credit to where it is due for the frameworks used while also highlighting existing issues with the frameworks not considered by the original authors, leading to more complete frameworks in the future.

This paper outlines the use of three frameworks in the design and development process of a set of serious games to teach computer architecture concepts. Those frameworks are: the Mechanics, Dynamics, and Aesthetics framework (Hunicke, LeBlanc, & Zubek, 2004); the Design, Play, and Experience framework (Winn, 2009); and the Serious Game Design Assessment framework (Mitgutsch & Alvarado, 2012). This paper discusses elements of serious game design and development encountered in the design and development process of the games, which frameworks accounted for those elements, and what elements the frameworks did not address. The result is a better understanding of how multiple frameworks can inform serious game design and development, a recognition of elements that existing frameworks cover well (mechanics, target audience, resulting experience, etc.), and a recognition of elements future frameworks need to address (polish, application context, resources, etc.).

Keywords: Serious Games, Educational Games, Serious Educational Games, Serious Game Design Research, Serious Game Frameworks, Digital Game Based Education

Introduction

The design and development process of serious games is a serious concern for those wanting to create serious games. Knowing the best design and development practices helps ensure that the most impactful serious games are made.

Serious game design frameworks provide explanations of important elements that need to be accounted for in the design and development process. However, a single framework often covers only a subset of all elements that need to be considered. Referencing multiple frameworks could alleviate this problem. This paper delves into the design and development process of a set of serious games informed through the referencing of three frameworks to examine if the use of multiple frameworks led to more elements being addressed.

The Frameworks

The frameworks referenced in the design and development of the games included: the Mechanics, Dynamics, and Aesthetics framework (MDA) (Hunicke, LeBlanc, & Zubek, 2004); the Design, Play, and Experience framework (DPE) (Winn, 2009); and the Serious Game Design and Development framework (SGDA) (Mitgutsch & Alvarado, 2012).

These frameworks were selected because they cover a wide range of concerns. MDA puts a focus on the relationship between the designer and the player (Hunicke, LeBlanc, & Zubek, 2004). DPE expands directly from MDA and identifies a broader set of concerns in the design and development process (Winn, 2009). SGDA puts its emphasis on the cohesion between the serious goal of a game and other elements (Mitgutsch & Alvarado, 2012). The full set of elements covered by the individual frameworks will become apparent in the later discussion of elements encountered when making the games.

The Computer Architecture Games

The computer architecture games made using the frameworks were created for a study comparing the application of a singular cohesive narrative to cover learning content in an educational game against the use of several games with varied narratives (no narrative tie between them) that cover the same content together. The preliminary results of applying the games were discussed at the Foundations of Digital Games Conference 2021 (McClintock & Owen, 2021). This paper instead focuses solely on the design and development of the games and highlights screenshots from the cohesive narrative version of the game.

Elements of Serious Game Design and Development Encountered

Elements both identified and not identified by the frameworks were encountered when making the computer architecture games. These elements break down into *in-game elements*, *design elements*, *development elements*, and *serious elements*. Each of these sets of elements influenced the design and development process of the computer architecture games. How well the utilized frameworks covered them varied. This coverage will be shown in tables such as **Table *1***. In all these tables, **D** means the framework directly accounts for an element, **I** indicates that the framework indirectly accounts for the element by incorporating it as a part of another element, and a blank space means the framework does not touch on the element at all.

In-Game Elements

In-game elements are elements that exist within the game and are primarily experienced directly by the player. While all in-game elements have to be designed by the designer(s) of a serious game, these elements are separated from the design elements by the fact that they are significantly viewed or interacted with from the player's perspective. In-game elements include: mechanics, narrative, audio, art, and game cohesion. **Table *1*** shows which frameworks discuss these elements.

Table 1

In-Game Elements Coverage in Frameworks

Framework	*Mechanics*	*Narrative*	*Audio*	*Art*	*Game Cohesion*
MDA	D	D			
DPE	D	D			I
SGDA	D	D	I	D	D

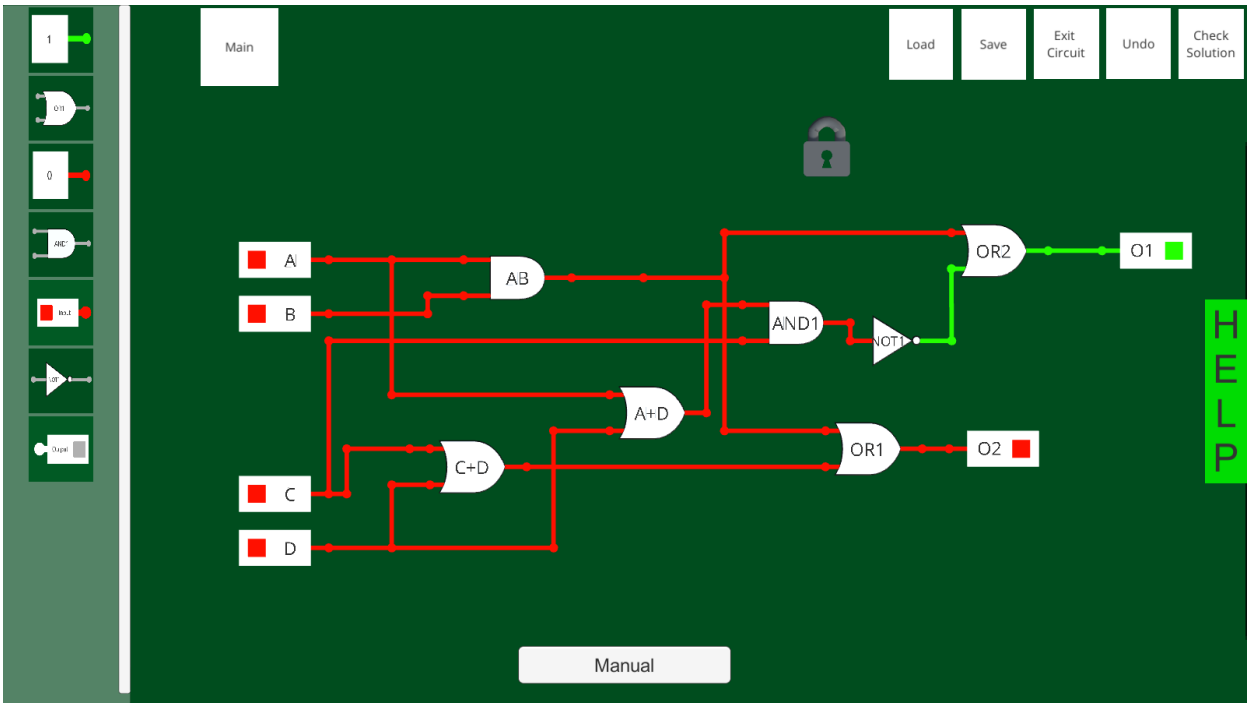
Mechanics

MDA, DPE, and SGDA all agree that mechanics are one of the most important elements of a serious game (Hunicke, LeBlanc, & Zubek, 2004) (Winn, 2009) (Mitgutsch & Alvarado, 2012). Mechanics are: "The rules that define the operation of the game world, what the player can do, the challenges the player will face, and the player's goals." (Winn, 2009) Mechanics are what the player does when interacting with the game. In the computer architecture games this includes: moving the player character around, talking to non-player characters, moving puzzle pieces, and so on.

The outlined importance of mechanics from the frameworks led to mechanics being a primary focus in the design and development of the computer architecture games. In particular, the frameworks push towards the alignment of mechanics with the purposeful goal of the games. This meant creating mechanics directly associated with the targeted learning content. Those mechanics are the interactive circuit puzzles in the game, which have more focus and depth than the 3D platforming mechanics used in the game's overworld to move the player between puzzles. An example puzzle is shown in **Figure *1***.

Figure 1

Circuit Puzzle in the Computer Architecture Games



Narrative

Narrative refers to both the embedded narrative, which is the narrative created for the game by the designer, and the emergent narrative, the story the player experiences through their playthrough of the game (Salen, Tekinbaş, & Zimmerman, 2004). In DPE, these are discussed as the designer's story and player's story respectively (Winn, 2009). This designer's story, or embedded narrative, is being included as a part of the narrative element within the in-game elements because it is something that the player interacts with and experiences. SGDA instead describes the narrative element's relation to the purpose of a serious game using examples to highlight how narrative should fit with purpose (Mitgutsch & Alvarado, 2012).

The frameworks outline narrative as, at minimum, providing the reasoning for the player's actions in the game and, at best, having a direct connection to the serious game's purpose. In the design of the computer architecture games, the narratives involved reached both of these goals. Following the guidance of the frameworks led to the narratives providing reasons for the player's actions in the game

while also sprinkling in important computer architecture concepts into parts of the dialogue between characters.

Audio

Audio refers to the sounds within the game that the player hears. This includes sound effects, background music, and any spoken dialogue. Only SGDA touches on the audio element as part of what it refers to as *aesthetics*, defined within SGDA as: "The audiovisual language conceptualized, chosen and developed by the designers for the visualization, and the display of the elements involved in the game." (Mitgutsch & Alvarado, 2012) Despite mentioning audio in the definition, SGDA's more detailed discussion of aesthetics focuses on the graphical part of their definition while leaving out audio.

Unfortunately, the computer architecture games lack audio due to resources constraints. However, audio was considered during the design and development process. In particular, development wanted to focus on getting audio tied into the circuit puzzles because the mechanics involved in those are directly related to the purposeful goal and that prioritization would be supported by SGDA.

Art

Art refers to the graphics and other visual elements of the game; everything the player sees. SGDA argues that art, as part of their definition of *aesthetics*: "Plays a fundamental role in the introduction of the game's purpose and its impact on the player." (Mitgutsch & Alvarado, 2012) SGDA also argues that art be related as much as possible to the purposeful goal of a serious game, as it does with all elements it covers.

What this meant for the design and development of the computer architecture games was that the art quality was kept similar across the games, so as to avoid art quality as a confounding variable in the research application of the games. It also meant that art in all the games was tied to the relevant narrative and educational purpose of the games.

Game Cohesion

Game cohesion refers to the intertwining of the other in-game elements with each other. Strong game cohesion occurs when the in-game elements work to support and enhance each other (inter-element cohesion) and themselves (intra-element cohesion).

An example of strong inter-element cohesion would be a low gravity jump mechanic for the player character in combination with a narrative that has communicated that the player is on the moon and why, audio that provides the matching sound of an astronaut jumping and unsettling dust on the moon, art that presents a convincing visual representation of the moon around the player character alongside a suitable astronaut model for the player.

Poor inter-element cohesion would be in-game elements that do not match up with each other; art showing the player on the moon, but a jumping mechanic that is the same as when the character was on Earth.

An example of strong intra-element cohesion in the art element would be a consistent visual style and color palette used throughout the game, such as maintaining a cartoonish style.

An example of poor intra-element cohesion in the narrative element would be a story with contradictions, plot holes, and inconsistencies that are never resolved.

Game cohesion is important for serious games and games in general. Strong game cohesion helps maintain immersion by providing the player with a world that maintains consistent rules that the player agrees to and exists in during play. Providing poor game cohesion risks a game that repeatedly breaks the player out of their immersion.

SGDA argues in favor of having more cohesiveness between the parts of a serious game and the game's purpose (Mitgutsch & Alvarado, 2012). SGDA argues through its example games that more cohesion strengthens a serious game by aligning the game's parts with its purpose, which should result in a higher likelihood of the game's purpose being realized (Mitgutsch & Alvarado, 2012).

SGDA's recommendations led to game cohesion being maintained in the computer architecture games by making sure all elements of the game related back to the purposeful goal and that the elements were consistent within themselves.

Design Elements

Design elements are elements that are considered primarily by the designer(s) of a serious game. The design elements include: polish, technology, user interface, target audience, balance, application context, gameplay/play dynamics, resulting experience, designer perspective, player perspective, academic perspective, content expert perspective, and design cohesion. **Table *2*** shows which frameworks discuss design elements.

Table 2

Design Elements Coverage in Frameworks

Framework	*Polish*	*Technology*	*User Interface*	*Target Audience*	*Balance*
MDA				I	I
DPE		D	D	D	D
SGDA				D	
Framework	***Application Context***	***Gameplay/Play Dynamics***	***Resulting Experience***	***Designer Perspective***	***Player Perspective***
MDA		*D*	D	D	D
DPE		*D*	D	D	D
SGDA			*D*		
Framework	***Academic Perspective***	***Content Expert Perspective***	***Design Cohesion***		
MDA					
DPE	*D*	*D*	*D*		
SGDA			*D*		

Polish

The idea of polish comes from the game development industry and is not mentioned in the frameworks. Polish is the finishing touches across the in-game elements. The game development industry refers to polish as the last ten percent of work (Bithell, 2015) (Russell, 2012). According to Naughty Dog's Benson Russell, polish is the removing of imperfections from the game that could take the user out of their immersion in the experience (Russell, 2012). Any of the in-game elements can be at differing states of polish.

A mechanical example of lack of polish would be a character controller that gets stuck on the geometry or shape of a level. This would cause frustration for the player that can no longer play the game because of a fault of the game and not of their own skill. This then leads to the player being removed from their immersion. Inversely, a mechanical example of polish would be a character controller that does not get stuck on the geometry of a level.

Minor inconsistencies in the story, missing sound effects or unsuitable sounds, or a texture that is slightly out of place are all examples of lack of polish. These may seem like insignificant issues when compared to their game breaking counterparts, such as a completely incoherent story, but the game development industry still puts focus on polishing their games to minimize the likelihood of a player's immersion being broken.

It is just as important in serious game design and development to resolve these smaller issues so that the player remains immersed in the experience, making the purpose of the game more likely to be realized.

While polish is not directly covered by the frameworks employed, it was still attempted in the computer architecture games. In the computer architecture games' development cycle, this meant resolving these smaller issues. Some of the more polished parts of the games include the overworld's animations when the player has made a working circuit puzzle, such as the seven segment display that

runs off of the player's built circuits as seen in **Figure *2***. Another example is a substantial amount of time ensuring the game physics did not result in the player becoming stuck.

Figure 2

Seven Segment Display Working from Player's Circuits

Technology

Technology refers to the hardware and software that the game uses. A designer needs to be cognizant of the limitations and affordances their chosen technology supports. For example, a Virtual Reality (VR) game gives the designer a unique control scheme to work with in which they can have the player physically move their body to provide input. This might work well in a serious game with the purpose of encouraging exercise which lines up with possible mechanics. However, the designer must also consider the limitations VR presents. VR set ups are expensive. They require a powerful computer and additional devices. If the serious game needs to reach a community without access to VR devices, then the designer has good reason to not choose to develop a VR game.

DPE summarizes the technology element well: “Overall, the capabilities and limitations of the technology and the resources required to implement the technology may greatly influence the design and should be considered throughout the design process” (Winn, 2009).

The framework’s recommendations on technology had a strong beneficial influence on the computer architecture games. The games needed to run on a course website. This meant that the games needed to run on web browsers. The design of the games was adjusted to work within the computational power limitations introduced by this. Because this was done early in the design process, the possible issues introduced by the technological limitations of web browsers were avoided and the advantages, easy distribution to students, were achieved.

User Interface

User interface (UI) "encompasses everything the user sees, hears, and interacts with and how that interaction happens (i.e., the control system.)" (Winn, 2009) This includes the physical interactive components of a game, such as a controller or keyboard, and the interactive screens the player sees in game, such as the manual used in the games shown in

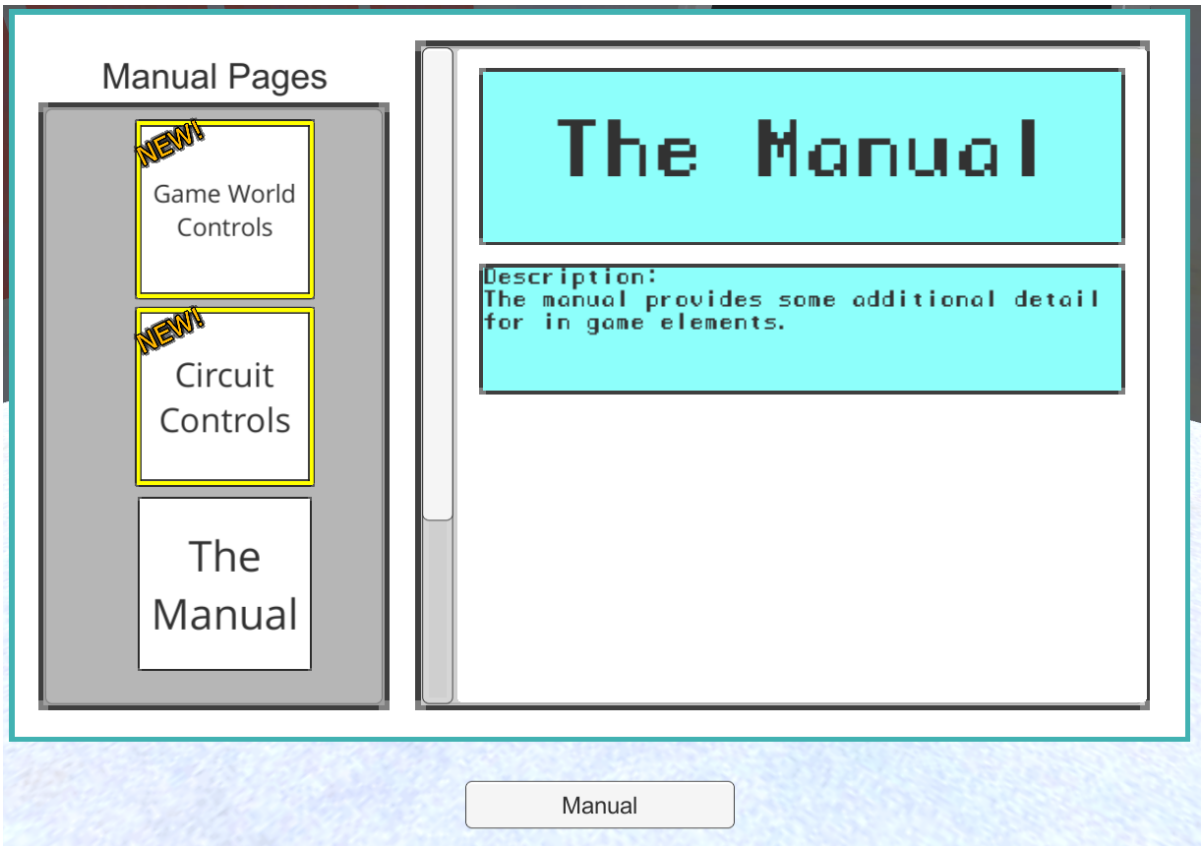
Figure *3*.

Good UI is easy for a user to miss, meaning they don't notice it because it is easy to interact with. In the computer architecture games, UI was considered in making sure the various buttons and drag-able pieces of the UI in the game were easy to interact with. This included making sure buttons in the same group, such as the scroll space of the manual pages on the left of the manual in

Figure *3*, were sized the same and spaced apart evenly.

Figure 3

Manual UI in Game



Target Audience

Target audience refers to who is meant to play the game. All of the frameworks discuss the target audience (Hunicke, LeBlanc, & Zubek, 2004) (Winn, 2009) (Mitgutsch & Alvarado, 2012). According to DPE: "Play is greatly influenced by not only the designer, but also the player, including his or her cognitive, social, cultural, and experiential background that he or she brings to the given play experience… The target audience for the game must be strongly taken into account throughout the design process." (Winn, 2009) Both DPE and SGDA point out that every player is unique and comes into playing a game with their own experiences and expectations (Winn, 2009) (Mitgutsch & Alvarado, 2012).

SGDA's example on how this can cause issues is play literacy, where different players might already be game players and thus pick up the controls quickly while others are not and will need the game to bring them up to speed (Mitgutsch & Alvarado, 2012). If the target audience is mostly new to gaming, the designer(s) must find a way to address that.

The recommendations around target audience had a strong beneficial influence in the design of the computer architecture games. The target audience for the games is computer science undergraduate students, typically third year students. This means that the target audience should already have a basic computer science background, include both game players and non-players, and potentially be culturally diverse. Accommodating this audience meant keeping the 3D platforming challenges in the game simple (to accommodate non-game players) while implementing a flow of challenging tasks in line with what the students should already know or not know, keeping the narratives approachable to a diverse audience, and lining up the flow of learning content in the game to fit into the course. Without the guidance from the frameworks on this element, many of the design changes that made the games more approachable would not have been made.

Balance

Balance refers to how well the game's challenges fit with the player's knowledge and skill. DPE relates balance to Mihaly Csiksentmihalyi's theory of flow (Winn, 2009). The original theory of flow focuses on pairing a skilled person with a challenging task that is neither too hard nor too easy (Csikszentmihalyi, 1990). This perfect matching of task to skill would put the person in a flow state in which they lose awareness of themselves, concentrate only on the task, and experience a sense of control (Csikszentmihalyi, 1990). DPE applies this to balance in games by pointing out that a game presents challenges at differing levels of skill requirements which can be matched to a player's skill level (Winn, 2009).

Early on in a game the challenges should have relatively low skill requirements to match the new player's weak skills. Then the game should maintain that balance of player skill over the course of the game. At any point in the game the challenge presented should provide the player with something they are ready for, but not something too easy such that it causes boredom for the player. Similarly, the challenges should never be so difficult that the player finds them frustrating. Balance in this sense is the

designer crafting a sequence of increasing challenges that always match with the player's skills, thus keeping the player in the flow state (Winn, 2009).

The computer architecture games followed these ideas from DPE on balance and flow. They present simple tasks at first to introduce how to move around the game world, how to do simple tasks in the circuit puzzles, and how to access the in-game manual. The games also follow the flow of learning content as presented in the course the games draw from and are designed for, balancing the learning challenges presented.

Application Context

The application context is where the game is being played. The place of play can be just as important to the designer as the experience and cultural background of the player. If the game is to be played in a live classroom, then the designer must consider the role of the teacher in the player's experience with the game. If the game is to be played as part of a fixture in a museum, then what does that mean for the game's design? Perhaps the museum will only allow each player to play the game for a set amount of time. The application context introduces design challenges that should be addressed by the designer. Unfortunately, none of the frameworks utilized covered application context. While this is not at all a fault of the frameworks themselves, it is important to consider that future serious game designers and developers might benefit from a framework including it alongside the other elements covered by the frameworks used or recognized in the process of making the computer architecture games.

The application context for the computer architecture games is a fully online classroom. This application context meant that the games would be played at home by students on various hardware and software.

This produced two possible options. One was to create operating system specific builds of the games for the most likely used operating systems of Windows, MacOS, and Linux. The other was to

produce WebGL builds that could be run on web browsers. WebGL builds were chosen so that the game could be hosted on the same website as a course which taught the same content as the games. This meant that players would not have to install the game on their systems to play it.

Gameplay/Play Dynamics

Gameplay/play dynamics refers to the interaction of the game's rules and in-game elements with the player's input. In MDA and DPE, dynamics sit in between the designer's created mechanics and the player's resulting emotional responses from the game play (Hunicke, LeBlanc, & Zubek, 2004) (Winn, 2009). By MDA's definition: "Dynamics describes the run-time behavior of the mechanics acting on player inputs and each others' outputs over time." (Hunicke, LeBlanc, & Zubek, 2004) In MDA, mechanics are focused on as what determines the dynamics (Hunicke, LeBlanc, & Zubek, 2004). In DPE's multi-layered framework, it accounts for an influence between its layers of learning, storytelling, gameplay, and user experience (Winn, 2009). In DPE a design decision on what content to teach as part of the learning layer exerts an influence on the mechanics designed in the gameplay layer which then has an effect on the dynamics.

DPE provides a deeper depth of understanding of how the dynamics could be influenced beyond just the designed mechanics. However, DPE does not fully include some of the in-game elements such as art and audio. In the definition of the gameplay/play dynamics element used in this paper, the interaction of ALL the mentioned in-game elements play a role, not just the mechanics.

Gameplay/play dynamics played into the design of the computer architecture games by keeping the designer mindful of how the various in-game elements would interact during play. For example, the art and mechanics elements interact when the player provides input to make their character jump. The character model animates in response to the input at the same time the game physically moves the character model up and down along the jump arc. Whether or not that interaction provided the player

with the desired resulting experience is the next consideration after that gameplay/play dynamic plays out.

Resulting Experience

The resulting experience element refers to the goals to be accomplished from playing the game. This is the desired end result for the player that the designer must consider. DPE and MDA include this as the resulting emotional response from the player (Hunicke, LeBlanc, & Zubek, 2004) (Winn, 2009). MDA refers to the player's emotional response as *aesthetic* which overlaps with the common use of the term to describe the visual and auditory parts of a game (Hunicke, LeBlanc, & Zubek, 2004).

DPE changes this terminology to *affect* which is a term meaning emotion or desire (Winn, 2009). DPE describes how the designer should consider the affective goals of their game, which are the emotional responses the game is trying to evoke (Winn, 2009). DPE provides example affective goals as the forms of fun defined by Heeter at al. (Heeter, et al., 2003). These forms of fun are beauty, immersion, intellectual problem solving, competition, social interaction, comedy, thrill of danger, physical activity, love, creation, power, discovery, advancement and completion, application of an ability, altruism, and learning (Winn, 2009) (Heeter, et al., 2003). DPE also expands on the resulting experience the designer should consider beyond the affective goal.

DPE includes "learning" as what the player learns from playing the game (Winn, 2009). Learning is one purpose a serious game could have. SGDA goes more broadly, arguing that design should revolve around the purpose of the serious game, which could be anything from learning to changing political opinions (Mitgutsch & Alvarado, 2012).

The recommendation from the frameworks led to a defining of the resulting experience and affective goals for the game which provided some influence over the design process. The resulting experience targeted by the computer architecture games was for the player to learn computer architecture concepts and find the games engaging. The affective goal for the resulting experience of the

games included the learning and intellectual problem solving forms of fun. Defining and then referencing these goals led to more focus on the circuit puzzles in the design and development process as they relate the most to the learning targeted by the games.

Designer Perspective

The designer perspective element refers to the view of the game from the eyes of the designer(s). A designer has a unique perspective on the game because of their background in game design. The perspective of the designer is: “Focused on creating engaging and entertaining game play” according to DPE (Winn, 2009). The designer is able to suggest ways in which the game can be made fun. They will know what in-game elements will fit best with the purpose of the serious game and lead to the desired resulting experience for the player.

Researchers working with serious games should carefully consider the designer perspective and ensure that a game designer is present when developing a game. Without the designer perspective the game risks losing the engaging interaction that is central to any successful game.

The designer must consider their own design perspective and be knowledgeable of holes in their own perspective. It is very rare for any one designer to have a depth of knowledge and background for every in-game element. Someone who has spent their career as an audio designer will have great input for the audio element but will likely have weaker input for the mechanics element. Likewise, a gameplay designer is likely to have great input for the mechanics but not audio. Filling the holes in a team’s design perspective is important to ensuring that a serious game is the best game it can be.

The designer perspective was considered in the computer architecture games by a game designer whose background includes the design of mechanics. Holes in the designer perspective of the team were filled in through conversations with Michigan State University game development faculty members knowledgeable in the relevant design spaces. Gathering more designer perspectives led to general improvements across the game, such as more concise platforming levels.

Player Perspective

The player perspective element refers to the player's unique view of the game. Target audience defines who the players are, whereas the player perspective defines how those players view the game and feel about it.

The designer(s) begin with an initial idea of what the player perspective will be based on the defined target audience for the game. The player perspective the designer(s) design around changes over the iterative design process when the designer(s) receive player feedback after playtesting.

The iterative design process is the cycle of *design*, *prototype*, *playtesting*, that repeats itself over and over again until the prototype game becomes the final release game (Winn, 2009). Players that fit the defined target audience playtest the prototype game and provide the designers with feedback, the real player perspective, which then influences the next iteration of design changes.

MDA and DPE argue the importance of considering the player's perspective (Hunicke, LeBlanc, & Zubek, 2004) (Winn, 2009). MDA puts it like this: "When working with games, it is helpful to consider both the designer and player perspectives. It helps us observe how even small changes in one layer can cascade into others." (Hunicke, LeBlanc, & Zubek, 2004). Under MDA's definitions, that means small changes to the mechanics lead to changes in the resulting experience for the player which can only be fully understood by considering the player perspective. This is expanded on in DPE where a change to any element can influence the player's experience (Winn, 2009).

In accordance with the guidance from using the frameworks, the player perspective was gathered for the computer architecture games from playtesting with the games' prototypes using players similar to the defined target audience. The advantages of considering the player's perspective included some changes made to make the tutorial parts of the game better spread out. Without considering this element as discussed in the referenced frameworks, the pacing of the game's would likely have been much worse.

Academic Perspective

Academic perspective refers to the researcher's view of the game. The researcher's perspective is unique in that they know the proper data collection and analysis methods that can be applied to answer the research questions the serious game is targeting. DPE states that the academic is: "Interested in various academic theories, be they from educational pedagogy, communication theory, etc." (Winn, 2009).

The designer and development team must consider the academic perspective during design and development to ensure that the resulting game is capable of answering the researcher's questions. This could involve adding data collecting methods directly into the game. The academic perspective element could also influence other elements. For example, the nature of the researcher's question could influence the technology that can be used. If the research questions are focused on virtual reality capabilities only, then the game must be made for virtual reality devices.

During the development of the computer architecture games the academic perspective led to the data collection methods integrated directly into the game. These methods included automatic (and anonymized) recordings of whenever a player edited a circuit puzzle, completed a circuit puzzle, completed a section of learning content (checkpoint), and so on.

Content Expert Perspective

Content expert perspective refers to the view from a person knowledgeable in the content area of the serious game's purpose, or as DPE puts it: "[the team member] interested in the given subject matter" (Winn, 2009). In a serious educational game, this could be provided by the instructor(s) familiar with the game's learning content.

Additionally, if the game is more specific in the community it targets, it should also employ a content expert on that community, preferable a member of the community, to provide their perspective to make the game's design more impactful for that community. In a 2019 paper on computer science

pedagogy, it was found that: “Teacher moves that encouraged students to share their personal perspectives almost always coincided with observations of increased student engagement.” (Ryoo, 2019) If a serious game can incorporate the cultural and experiential backgrounds of its target audience, then it should do so.

For the development of the computer architecture games, the content expert perspective was provided by a previous professor for the class the games were based on. This same professor had also previously created course materials that teach the same content that is in the games. The content expert perspective provided covered both the learning content the game targets and the community the game was designed for. The inclusion of this content expert perspective resulted in a design better aligned to the learning content.

Design Cohesion

Design cohesion refers to how well the various design elements, including the design of the in-game elements, work together. One example of design cohesion is the alignment of the various perspectives mentioned.

DPE calls the aligning of the academic, designer, and content expert perspectives the *heart of serious game design* (Winn, 2009). DPE argues that properly aligning those three perspectives leads to a whole that is greater than the sum of its parts (Winn, 2009). SGDA similarly pushes for cohesiveness in serious games as a whole, arguing that cohesion supports the purpose of the serious game (Mitgutsch & Alvarado, 2012).

DPE also discusses how its various layers can influence each other: “Certain design decisions are complementary or conflicting across the layers.” (Winn, 2009) A designer should seek to maximize design cohesion by recognizing the complementary aspects of their design decisions and by seeking them out. At the same time, the designer should attempt to minimize the conflicts across the design

elements. For example, choosing an art style that requires a cutting edge graphics card while the application context limits the technology to mobile devices is a conflict.

The design cohesion element as identified by the frameworks was considered across many design decisions in the development of the computer architecture games. For example, the content expert perspective and mechanics element align in that the mechanics are modeled off of what would be taught in class. Similarly, the balance of challenges and progression was modeled off of the content expert perspective. Overall, the consideration of design cohesion led to the games having a more complementary design.

Development Elements

Development elements are elements of the game design and development process that fall primarily under the view of the developer(s), which can also be the designer(s) for the game. The development elements are focused on the process of making the game. The design elements focus on the thoughts and expectations of the designer. The development elements then take those thoughts and turn them into the in-game elements. The in-game elements can then confirm or refute the designer's expectations through the use of playtesting, after which the process repeats itself until the best possible game is created. The development elements include: playtesting, iterating, prototyping, resources, asset acquisition, communication, and extensibility. **Table *3*** provides a breakdown of which frameworks discuss development elements.

Table 3

Framework	*Playtesting*	*Iterating*	*Prototyping*	*Resources*
MDA	D	D		
DPE	D	D	D	
SGDA				

Framework	*Asset Acquisition*	*Communication*	*Extensibility*
MDA			
DPE			
SGDA			

Development Elements Coverage in Frameworks

Playtesting

Playtesting refers to the act of playing the game to discover issues with the game that need to be addressed. These issues can be bugs, which are ways the game is not working as intended such as the game crashing. But these issues can also be problems with the resulting experience, such as when a play tester finishes playtesting the game and finds it unengaging or unimpactful to its purpose.

Both MDA and DPE cover playtesting. Playtesting is mentioned in MDA as part of their *tuning* process in which changes are made iteratively to better realize the design goals and remove flaws (Hunicke, LeBlanc, & Zubek, 2004). DPE covers it as part of its description of the *iterative design process* in which the game is designed, a prototype developed, the prototype play tested for feedback, and then the design changed and the process repeated (Winn, 2009). The key point to play testing is that it provides feedback for the team on whether the game is functional and whether design goals or the purpose of the game are being realized.

Playtesting can be done internally by the developers and designers to search for bugs that need to be fixed and to get an idea on whether the game is meeting its serious purpose. However, playtesting must be done with the target audience to get an accurate assessment of the resulting experience and purpose of the game at any state. Ideally this includes both first time play testers and repeat play testers from the target audience.

First time play testers are important because they will give the team an idea of how the game would be received by the target audience if it was released as is. Repeat play testers are important because they will develop a better understanding of the game over repeated sessions and thus provide deeper feedback over time.

Playtesting in the development process of the computer architecture games was done internally by the designer and externally with players (first time and repeat) resembling the target audience. The advantages of playtesting included the catching of various bugs during development and feedback from players leading to smoother tutorial parts in the games.

Iterating

Iterating refers to the part of game development in which changes are made to the game based on the feedback received from playtesting. This includes the fixing of bugs and the changing of the in-game elements and design elements in an effort to better achieve the purpose of the serious game and improve the resulting experience. In MDA this is referred to as *tuning* (Hunicke, LeBlanc, & Zubek, 2004) and DPE encapsulates iterating within its *iterative design process* (Winn, 2009).

In the computer architecture games, iterating and the iterative design process as described by DPE were used to keep the games on track to achieve their purpose and be engaging. In general, this made the design and development run smoothly.

Prototyping

Prototyping refers to the creation of a feature incomplete version of the game that is meant to be play tested for feedback that will then inform the iterations made. A prototype can be made to target specific areas of desired feedback and need not be digital. A paper prototype for example is a prototype created using physical media like paper and dice (Winn, 2009). The advantage of making a paper prototype is that a designer can test their design, or an approximation of it, without resources needing to be spent developing the digital version.

Prototyping, iterating, and playtesting should be done at all stages of a game's development. More complex prototypes can be made as development on the game is under way and assets are created. For example, a first playable prototype showing the mechanics of the game can be made by the programmer(s) utilizing programmer art (cubes and other primitive shapes). This prototype needs only the programmer's code that creates the desired mechanics, and its purpose is to gather feedback on those mechanics.

Once the various assets the game needs are created and start being implemented into the game, the game itself becomes a prototype to be play tested. At this point in development the game prototype should be kept in a play-testable state.

A play-testable state means the game has no game breaking bugs, such as those that cause a crash. Doing this allows play testing to provide the most relevant feedback on the game possible. Not doing so means that the version of the game ready for play testing does not resemble the current state of the game's development. Which means that feedback received from play testing is less applicable to the current development state.

The computer architecture games utilized prototyping early in development to show basic functionality of the game's mechanics, such as the dragging and dropping of puzzle pieces in circuit puzzles. The games were also maintained in play-testable states throughout their development. This meant that the games were play tested at all points of the process, leading to a smooth process of finding and fixing bugs that might not have happened without the iterative process pulled from DPE.

Resources

Resources refers to the people, skillsets, money, time, and other resources available to be used in development. The available resources determine what designs are feasible. If the skillset of the team is not complimentary to a design, then that design can be discarded. Similarly, if a design is too costly in

money, time, or any of the available resources, it becomes unfeasible. A consideration of resources is not discussed by the frameworks referenced in making the games.

While the resources element and its imposed restrictions should be left out of the initial brainstorming of game ideas, it is valuable throughout the rest of development. In the remainder of the brainstorming process the restrictions imposed by resources will quickly cut out unrealizable ideas. This saves the team from spending resources on an idea that will likely never come to fruition. Beyond brainstorming, the resources element acts as a check on whether the game's development is still on track for completion. Either more resources need to be collected or cuts to the design need to be made if the resources are being depleted too quickly.

The resources element imposed large restrictions on the development of the computer architecture games. The budget for developing the games was non-existent. The available skillsets to the team were programming, research, and teaching related. This meant that availability of art and audio assets were slim to none. Because of these limitations, resources were carefully allocated and the design changed to ensure the best possible games were still made.

Asset Acquisition

Asset acquisition refers to the purchasing, creating, or otherwise obtaining of assets to be used in game. These assets include music, sound effects, 3D models, sprites, code, levels, etc. The asset acquisition element requires the developer(s) to consider how assets are going to be acquired for the game. Will they be created internally or purchased? Regardless of how assets are acquired, they are essential building blocks for the game and therefore their obtainment is a serious concern. MDA, DPE, and SGDA do not discuss asset acquisition or how it might relate to the elements they do cover.

Asset acquisition requires the developer(s) to consider the resources they have available. Specific assets are more easily created with complimentary skill sets. A trained 3D artist will have an

easier time making a quality 3D model than a sprite sheet. If the skillsets the team needs for a design are not available, then the developer(s) must consider how their other resources can fill that gap.

Money can be used to buy existing assets off of stores such as Unity's Asset Store (Unity, 2022) or Turbosquid (Shutterstock, 2022). But finding existing assets that fit in well to the design of a game can be difficult. The developer(s) must manage their resources to keep development flowing smoothly, and they must coordinate with the designer(s) on limitations of resources because that availability will affect the design process. If certain assets cannot be acquired, then the designs requiring them cannot feasibly be completed.

Asset acquisition in the development of the computer architecture games was limited by the available resources. While all of the code was made by the team, the lack of art related skillsets meant that little art could be made by the team. This issue was resolved by reaching out to collaborators willing to allow the use of their work. The lack of audio in the games unfortunately could not be resolved the same way, as fitting assets that could be obtained freely were not found for all the places audio would be needed in the games' designs.

Communication

The communication element refers to the building and maintaining of relationships as well as the transfer of information between team members. Properly building and maintaining the relationships between team members ensures a smoother development process, as it eliminates conflicts between team members. The employed frameworks unfortunately did not have recommendations for communication as they are focused on other sets of topics.

Communication is an important aspect to every team based project, including serious game design and development. A serious game cannot be made easily without being able to effectively communicate the research goals, game design ideas, and purpose between team members. Similarly, the asset creation process of a game is hampered by lack of communication.

Games have many pieces, typically developed by different people. In order to put the pieces together, the artists, programmers, designers, and so on need to be in tight communication. A serious game developed with strong communication is more likely to have good game cohesion and design cohesion.

Tight communication was maintained during the development of the computer architecture games. The development team met at least once a week and utilized the task management software Trello (Atlassian, 2022) to keep track of development. This made it easier for the team to make changes and cuts throughout the development process.

Extensibility

Extensibility refers to how easy it is to build upon the project or any of its parts. Extensibility is important to consider because any task that is finished using an extensible solution will make new related tasks easier to complete. The frameworks referenced do not discuss extensibility.

An example of extensibility from the coding discipline would be creating scripts following a coding standard. A coding standard is a set of rules for programmers on a team that outlines how they should write code. This includes rules on how to write comments such as removing any large blocks of commented out code. Other rules in a coding standard can include how to name variables and classes. Coders on a team are able to easily work on each other's scripts if they all follow the same coding standard. This then improves extensibility as any new task related to a script can be picked up and done more easily by any programmer, not just the original creator.

This applies more generally. Every discipline has its own set of best practices that should be followed. Making sure the team follows best practices is a good way to improve extensibility on a project. However, there are threats to maintaining extensibility.

Perhaps the biggest threat to extensibility is simply the availability of resources and their proper application. When the team is put under a difficult deadline the likelihood of best practices being

followed decreases, or it is decided that best practices be dropped purposefully in favor of faster yet more error prone development. The solution to this is to manage resources effectively. However, these situations sometimes become difficult to avoid given that the iterative nature of game design and development can produce unforeseen problems.

Beyond extensibility inside of development, there are also ways to provide extensibility beyond the end of development. One way is to package the completed game in a publicly available repository. That way the game can be used again in future research. A better way is to package the entire game project, including relevant documentation and assets, so that future researchers can both use the game and make changes to it. Doing either of these things provides a way for a serious game to keep making an impact beyond its original application.

Extensibility in the computer architecture games included establishing a coding standard prior to the start of the games' development. However, extensibility issues occurred due to tight deadlines leading to code not perfectly in line with the standard. As for extensibility beyond the development, a release of the games' project files will be done when research using the games is completed.

Serious Elements

The serious elements are elements unique to serious game design and development. The academic perspective and content expert perspective elements could fit underneath this category, but they have been put under the design elements as those two perspectives are of most concern to the designers. This also allows more emphasis to be put on the element most unique to serious game design and development, the purpose element.

Purpose

Purpose refers to the impact the serious game is trying to have on the player. Because purpose is the primary point of making a serious game, it exerts influence over all the other elements. SGDA is the only framework to directly discuss purpose.

As SGDA puts it: “The driving force that functions as the pivotal influence over the elements of the game design should be the purpose of the game.” (Mitgutsch & Alvarado, 2012) This view is briefly reflected in the other frameworks in how they all argue that mechanics should be tightly coupled with the purpose of the game (Hunicke, LeBlanc, & Zubek, 2004) (Winn, 2009). Altogether, these frameworks make it clear that a serious game’s purpose should be the dominating concern in any design decisions. Similarly, purpose should govern development decisions.

The clearest example of how purpose influences development is in the loop of prototyping, playtesting, and iterating. The purpose of the game needs to be considered during playtesting to accurately judge whether the purpose is being achieved or not. Then that purpose relevant feedback is incorporated into the changes made in iterating which then culminates in the next prototype.

The purpose of the computer architecture games is to teach undergraduate computer science students computer architecture concepts. This purpose influenced every decision made in the design and development of the games, and is why much of the development focus, time, and resources were put towards the circuit puzzles as they most closely relate to the learning content, and this would follow the recommendation of the referenced frameworks.

Conclusions

Using multiple frameworks to inform the design and development process of a set of serious games led to more elements in those processes being adequately addressed than if only a single framework had been referenced. The frameworks were particularly useful in providing the iterative process from DPE which combined with the discussion of cohesion from SGDA led to a design for the games that emphasized their learning goals primarily through mechanics and supported by the other elements. However, there are still elements found during those processes that the frameworks did not account for which had influence over the design and development of the games: polish, application context, resources, asset acquisition, communication, and extensibility.

Some of those missing elements exerted a large influence on the process for the computer architecture games. Application context, resources, and asset acquisition in particular put limitations on the design. This included pushing the technology the game was built on to be web browsers and forcing the use of whatever art could be obtained at no cost. It is not the fault of the frameworks for not discussing these elements, as the frameworks are made to focus on specific topics. However, it is worth noting that even with three frameworks referenced in the design and development process, there were still important elements encountered that perhaps should be incorporated into future frameworks.

Future Work

One thing that might be beneficial for future serious game developers is the formation of a more comprehensive framework that touches on all the elements recognized in the field. Existing frameworks focus on individual elements or small sets of elements in the serious game design and development process. Combining the focuses across frameworks into a single framework would provide future developers a single point of reference to work from. This would be advantageous to the field as it would provide newcomers a strong place to start that references the other older frameworks the researcher should also review. Taking this a step further, this comprehensive framework could pull not just from other frameworks, but also from methodologies such as the Gameplay Loop Methodology (Czauderna & Guardiola, 2019) or from the many fields interested in serious games. Doing so would help make a robustly applicable serious game design and development framework that could also point to the various practical methods and solutions for addressing elements.

References

Atlassian. (2022). Trello. Retrieved from https://trello.com/

Bithell, M. (2015). Indie Polish: Making the Most of the Last 10%. *Game Developers Conference.* Mike Bithell Games. Retrieved from https://gdcvault.com/play/1022080/Indie-Polish-Making-the-Most

Csikszentmihalyi, M. a. (1990). *Flow: The psychology of optimal experience* (Vol. 1990). Harper & Row New York.

Czauderna, A., & Guardiola, E. (2019). The gameplay loop methodology as a tool for educational game design. *Electronic Journal of e-Learning, 17*(3), 207--221. Retrieved from https://academic-publishing.org/index.php/ejel/article/view/1884

Heeter, C., Chu, K., Maniar, A., Mishra, P., Egidio, R., & Winn, B. (2003). Comparing 14 forms of fun (and learning and gender issues) in commercial versus educational space exploration digital games. *International Conference on Digital Games Research.*

Hunicke, R., LeBlanc, M., & Zubek, R. (2004). MDA: A formal approach to game design and game research. *Proceedings of the AAAI Workshop on Challenges in Game AI*, *4*, p. 1722. San Jose, CA.

McClintock, D., & Owen, C. (2021). Common Narrative in Educational Video Games: A Design of Games to Teach Circuits. *Proceedings of the 16th international conference on the foundations of digital games*, (p. Publication Pending).

Mitgutsch, K., & Alvarado, N. (2012). Purposeful by design? A serious game design assessment framework. *Proceedings of the International Conference on the foundations of digital games*, (pp. 121--128).

Russell, B. (2012). The Last 10: Going From Good To Awesome. *Game Developers Conference.* Naughty Dog. Retrieved from https://gdcvault.com/play/1015601/The-Last-10-Going-From

Ryoo, J. (2019). Pedagogy that supports computer science for all. *ACM Transactions on Computing Education (TOCE), 19*(4), 1--23.

Salen, K., Tekinbaş, K. S., & Zimmerman, E. (2004). *Rules of play: Game design fundamentals.* MIT press.

Shutterstock. (2022). Turbosquid. Retrieved from https://www.turbosquid.com/

Unity. (2022). Unity Asset Store. Retrieved from https://assetstore.unity.com/

Winn, B. M. (2009). The design, play, and experience framework. In *Handbook of research on effective electronic gaming in education* (pp. 1010--1024). IGI Global. Retrieved from http://ksuweb.kennesaw.edu/~rguo/2015_Spring/CGDD4303/readings/winn-dpe-chapter.pdf