Live Game Design: You Make the Rules

Riemer van Rozen
rozen@cwi.nl
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands

ABSTRACT

Designing games is difficult and time-consuming. To speed up and simplify the design process, we have developed the Vie app [4]. Vie is a visual programming language for quickly creating 2D game prototypes [2, 3]. After a brief introduction, you can create your own game rules and instantly see your ideas come to life. To get you started, you will receive assignments and a handy cheat sheet.

ASSIGNMENTS

There are two assignments, an easy one and a difficult one. We use the Machinations language to express the rules [1]. Using its notation, we create a digital marble track that works inside a game.

ASSIGNMENT 1: VIE AND HER BUNNY

We begin with an example about a girl who lost her bunny. Her name is Vie, just like the app. Figure 1 shows the images: Vie, Bunnies, Apples and Heart. Step by step, we will use these elements in the game's design. You can play it immediately.

Step 1: Vie joins in the game

Step 1.1. **Goal.** We begin by adding Vie to the game.

Action. Start the Vie app. Below Machinations (top left) you can see a circle. Drag the circle and drop it on the center of the screen.

Result. There is now a circle on the screen. In Machinations, this is called a *pool*. It is a place where resources (marbles) can be stored. If there is a marble inside, it means Vie is there.

Step 1.2. **Goal.** We will make sure that there is just one Vie.

Action. You can use the Node Editor (bottom left) to adjust the pool. Set the values of the At and Max fields to 1.

Result. When you click in the UI Design or Game tabs, you will see that a Sprite (a picture) and a Label (name and value) have been added. When Vie is there, you will see her picture.

Step 2: Vie leaves for a moment

Step 2.1. **Goal.** We add a rule called Leave for "going away".

Action. Below Machinations (top left) you can also see a triangle that points down. Drag and drop the triangle to the right of Vie. In the Node Editor (bottom left), set the value of When to "User". The Leave rule will become an interactive button in the game.

Result. There is now a triangle on the screen called a *drain*. This is a place where marbles can disappear. From the double lines, you can tell that Leave is an interactive game element.

Step 2.2. **Goal.** Marbles need a path to roll along. We will add a connection so Vie can leave.

Action. Click on the dot on the right of the Vie pool, and connect the line with the dot on the left of the Leave drain.

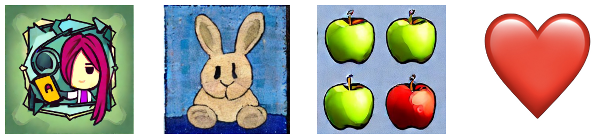
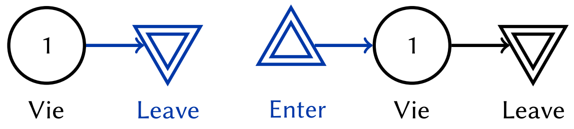

(a) Vie (b) Bunnies (c) Apples (d) Heart

Figure 1: Game Elements



(a) Rules after Step 2 (b) Rules after Step 3

Figure 2: The first versions of the rules

Result. There is now a line between Vie and Leave. This is called a *resource connection*. The rules now appear like Figure 2a. We have added the blue elements in Step 2. A Leave Button has also been added. If you click that button in the UI Design or Game tabs, Vie will go away. You can try it out.

Cheating is also allowed. You can also load the answer to Step 2. Click Menu, select "1. Vie (step 2)", and click Load.

Step 3: Vie returns on stage

Step 3.1. **Goal.** We will add an Enter rule for "returning".

Action. Below Machinations (top left) there is also a triangle that points up. Drag this triangle and drop it to the left of Vie. In the Node Editor, set When to "User". This also adds a button.

Result. There is now a second triangle on the screen. This is called a *source*, a place where marbles can come from. From its double lines, you can tell Enter is an interactive source.

Step 3.2. **Goal.** We will add a connection that lets Vie return.

Action. Click on the dot on the right side of the Enter source, and connect the line with the dot left of the Vie pool.

Result. There is now a line between Enter and Vie, another resource connection. The rules now appear as in Figure 2b. We have added the blue elements in Step 3. If you click on the Enter button in the UI Design or Game tabs, Vie will return. Try it out.

Cheating is also allowed. You can also load the answer to Step 3. Click Menu, select "1. Vie (step 3)", and click Load.

Step 4: Vie always brings an apple

Step 4.1. **Goal.** We will now add Apples to the game.

Action. We again start with Machinations (top left). Drag another circle to a good spot on the screen, e.g., below Enter.

Result. There is now a second pool on the screen. If there are marbles in this spot, these are Apples. A Sprite and a Label have been added in the UI Design and Game tabs.

Step 4.2. **Goal.** Every time Vie returns, she brings an apple.

Action. Click on the dot on the right side of the Enter source, and connect the line with the dot on the left side of the Apples pool. Next, click on the Enter source. In the Node Editor, set the value of How to "All" to ensure Vie only brings an Apple when she returns.

Result. There is now a resource connection between Enter and Apples. If you now click on the Enter button in the UI Design or Game tabs, then Vie brings an apple when she returns. Try it out.

Cheating is also allowed. You can also load the answer to Step 4. Click Menu, select "1. Vie (step 4)", and click Load.

Step 5: Vie swaps the apples for her bunny

Luckily, Vie can get her bunny back for exactly four apples.

Step 5.1. **Goal.** We will add Bunnies to the game.

Action. We again start with Machinations (top left). Drag another circle and drop it in suitable place on the screen. In the Node Editor, adjust the value of Max to 1 so there can only be one bunny.

Result. There is another pool on the screen. When there are marbles inside, these are Bunnies. In the UI Design and Game tabs, a Sprite and a Label have been added to see them.

Step 5.2. **Goal.** We will add a Swap rule and a Swap button.

Action. Below Machinations you can also see a triangle pointing to the right with a vertical line through it. Drag it to a suitable spot.

Result. There is now a new element on the screen. Swap is a converter. Converters can exchange one kind of resource (marbles) for another kind. Because converters are interactive, a Button has also been added in the UI Design and Game tabs that activates it.

Step 5.3. **Goal.** We add that the bunny costs four apples.

Action. Add a resource connection between Apples and Swap. Click just above the connection, and enter amount 4.

Result. The new source connection indicates that the cost of trading is 4 apples. However, swapping has no benefit yet.

Step 5.4. **Goal.** We add that for swapping we get the bunny.

Action. Add a connection between Swap and Bunnies.

Result. Now we can exchange apples for the bunny. Click on the Swap converter, or on the Swap button in the UI Design or Game tabs. If you have enough apples you will get the bunny.

Step 6: Happily ever after – or something else!

A game is never completely finished, not even Figure 3. Add rules yourself and try them out. Let a Heart appear, or make up rules about Bananas or Cows. Maybe they're hungry!

ASSIGNMENT 2: CLIMATE CHANGE

You can also design complex games with Vie. In this exercise you will make an educational game about climate change.

Goal. Design a game using the elements: Heat, Factories, Trees and CO2. What can you do to prevent global warming?

Action Click Menu, select theme "The Climate" and click Done. Design rules and try them out. Tip: discuss and play!

Example. You can also load an example. Click Menu, select "2. Climate", and click Load. This is not a very fun game yet. Can you make better rules for improving it?

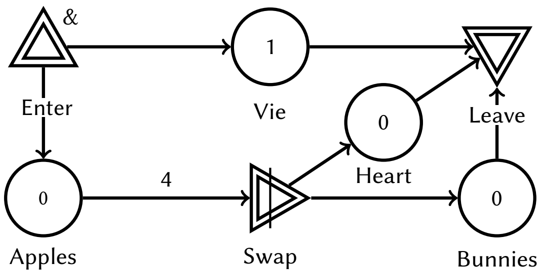


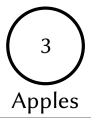
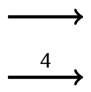
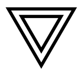
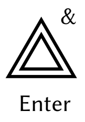
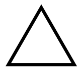
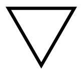
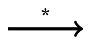
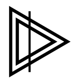
Figure 3: A complete version of the rules

REFERENCES

[1] Ernest Adams and Joris Dormans. 2012. *Game Mechanics: Advanced Game Design.* New Riders.
[2] Riemer van Rozen. 2023. Cascade: A Meta-Language for Change, Cause and Effect. In *International Conference on Software Language Engineering (SLE 2023).* ACM.
[3] Riemer van Rozen. 2023. Game Engine Wizardry for Programming Mischief. In *International Workshop on Programming Abstractions and Interactive Notations, Tools, and Environments (PAINT 2023).* ACM.
[4] Riemer van Rozen. 2024. Vie app – v0.0.5. (Oct. 2024). https://vrozen.github.io/Vie

CHEAT SHEET

Machinations is a visual language for designing a game's rules. Diagrams (or programs) are graphs that consists of two kinds of elements: nodes and edges. Both can be adjusted with extra information. These elements determine how resources (marbles) are step by step redistributed (roll) along the paths of the diagram (the marble track). This cheat sheet describes the maint language elements.

	A *pool* is a node with a name that can contain resources (marbles) such as coins, crystals or apples. A pool appears as a circle with a number inside that represents the current amount, and the starting amount (at). The maximum capacity (max) determines when a pool is full, and no more marbles can be added.
	A *resource connection* is an edge with an associated amount that represents the rate at which marbles roll between source and target nodes. Every step, each node can work once by redistributing marbles along the resource connections of the marble track. The inputs of a node are the resource connections on the left, and the outputs are on the right.
	The *activation modifier* (when) determines when a node can work. By default, nodes are *passive* (no symbol). *User* nodes (double line), represent interactive elements offering actions users can activate. Automatic nodes (*) work automatically.
	Nodes work (act) either by *pulling* marbles along their inputs (default, no symbol) or by *pushing* marbles along their outputs (p). They work in two ways (how). Nodes that have the *any* modifier (default, no symbol) interpret the amounts of their source connections as upper bounds and move as many marbles as possible. For nodes that instead have the *all* modifier (&), these are strict requirements, and the associated flows either all happen or not at all.
	A *source*, a node shown as a triangle pointing up, is the only element that can generate marbles. Sources can be seen as a pool with an infinite amount of marbles. They can always provide sufficient marbles.
	A *drain*, a node shown as a triangle pointing down, is the only element where marbles can disappear. Drains can be seen as pools with an infinite negative amount of marbles. They can always consume more.
	A *trigger* is a connection whose value is a multiplication sign (*). The source node of a trigger activates the target node when sufficient marbles roll for each resource connection on which that node acts.
	Converters are nodes shown as triangles pointing to the right with a vertical line through the middle. They consume one type of resource and produce another. Converters only work if all the required marbles are available at the inputs.