Lazarus: Stages 5, 6, & 7

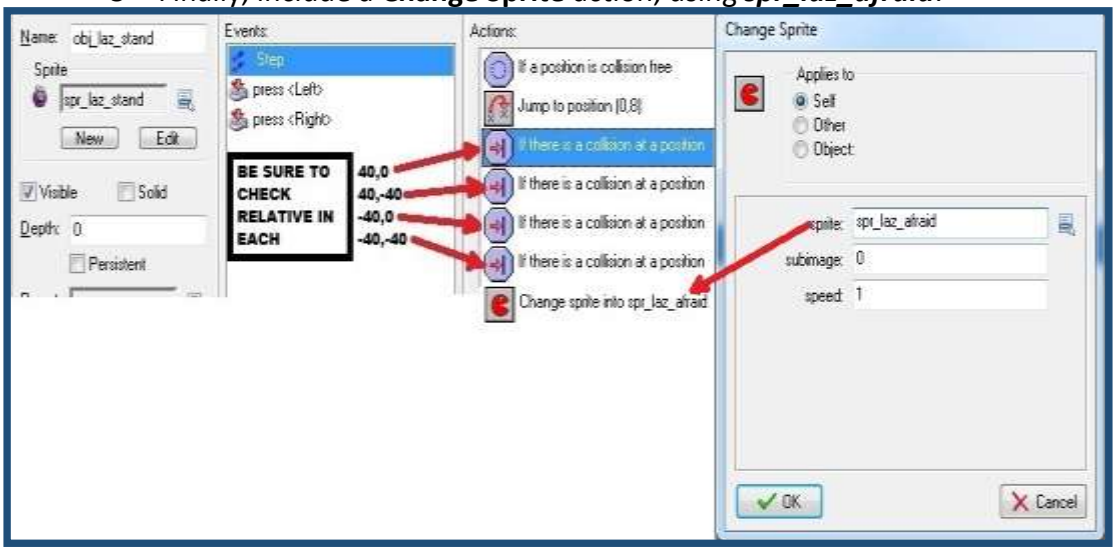
Of the game builds you have done so far, Lazarus has had the most programming properties. In the big picture, the programming, animation, gameplay of Lazarus is relatively simple. Planning and organizing your programming steps becomes more important as game properties become more numerous. You have seen many examples of how events, actions, conditional actions, and blocks come together to create more complex logic systems. Game developers, programmers, and other professionals in the field, rely on many processes and steps in order to make ideas into working playable products. The process includes stages for **concept development** and **game development**, like writing a **concept document**, **storyboarding**, **asset development**, **flowcharting**, and **prototyping**. If there are technical problems in the programming, called **bugs**, then programmers must **troubleshoot** (**debug**) the program. Once the game is **prototyped**, it can be tested by people outside of the design process. We will study *alpha* and *beta testing* in a separate activity in the future. Let's recall the *six step design process* studied at the beginning of this course. All phases of **concept** and **game development** for video games fit somewhere in that process, as you will see the *six step flow map* again these stages. The story behind Lazarus, his behaviors, the *physics*, *gameplay*, and the *animation* all start with an idea. As you put the finishing touches on Lazarus, you will read about the process of **game development**, and see examples of just a few tasks and stages that happen in the course of **game development**. You will soon be challenged to develop your own game. In that experience, you will get a taste of what it takes to design a game. Start by adding the new vocabulary meanings to your tutorial guide.

- **BE SURE TO COMPLETE HYPOTHESIS STATEMENTS FOR EACH STAGE.**
- **DO NOT USE THE TEST BUTTON IN THIS ACTIVITY UNTIL THE END!**
- **OPEN YOUR FILE YOU SAVED AS *initials_lazarus***
- **THERE ARE NO SAVE PROMPTS IN THIS TUTORIAL, SO SAVE REGULARLY.**

STAGE 5:

More editing of the Lazarus object…….What makes him show fear?

1- Reopen the ***obj_laz_stand*** object and then reopen its **Step** event, so that you can see the existing actions for this event.
2- Include the **Check Collision** conditional action (found in **control** tab) below the last action in the list. Set **X** to **40** and **Y** to **0**, and enable the **Relative** option.
3- Include another **Check Collision** action with **X** set to **40**, **Y** set to **-40**, and the **Relative** option enabled. This checks for a box diagonally to the right of Lazarus.
4- Include two more **Check Collision** actions: one with **X** set to **-40** and **Y** set to **0**, and the other with **X** set to **-40** and **Y** set to **-40**. Both should ha the **Relative** option enabled.
NOTE: These check for boxes to the left and diagonally up to the left.
5- Finally, include a **Change Sprite** action, using ***spr_laz_afraid***.

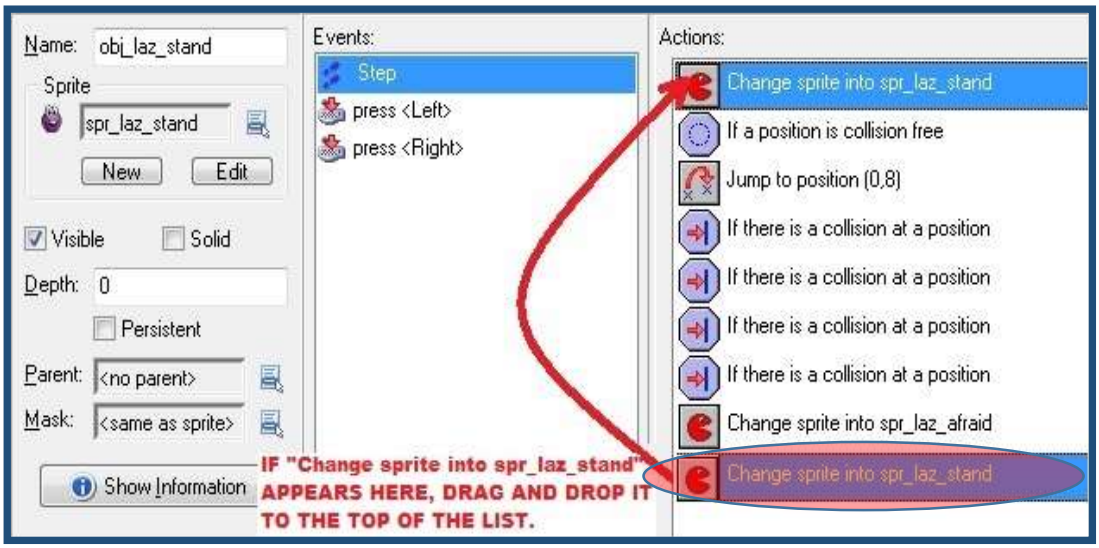


Concept Development

The process of designing video and computer games involves many tasks. Popular games that are sold to consumers involve hundreds of people, each with specialized roles that are required to make the idea, or concept, into a final game. One phase is **concept development**. This includes many non-technical tasks, such as brainstorming, writing, storyboarding, and any other tasks that allow the concept, to be communicated, planned, and prepared for the technical and programming tasks. Two tasks of high importance are the writing of **concept documents**, and **storyboarding**.

1- If you closed the form for ***obj_laz_stand***, go ahead and reopen it. Select the **Step** event again to see the existing actions for this event.

2- Note that this next action must be placed at the top of the list. Add a **Change Sprite** event and set it to change into the ***spr_laz_stand***. This action must be at the beginning of the list for this step event, so **if it is at the end of the list, move it by dragging and dropping it to the top of the list**. **It <u>cannot</u> be at the bottom of the action list.**

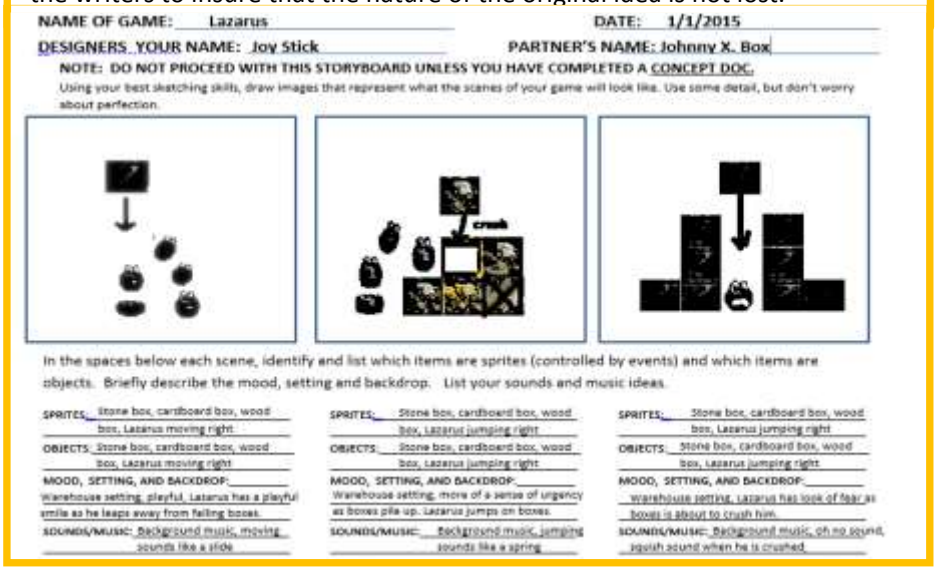


<u>Concept Document</u>

During **concept development**, writers and producers will meet to discuss, or *brainstorm,* ideas that will affect the future of the game project. Everything form the plot, or storyline, characters, gameplay, and what the game will look like, must be carefully considered. They will consider the target group of people will be that might be interested in buying the game. A written report, called a **concept document**, must be prepared. It details an idea for a game that might be developed. The **concept document** might be used to convince producers to commit time and money into the development the idea. A short **concept document** sample is shown below.

<u>Storyboarding</u>

Another crucial task done in concept development is **storyboarding**. This involves the drawing of different frames, each representing an event, action, or level in the game. It allows the images to grow out of ideas into a two dimensional drawing form. A storyboard shows the look of characters, backgrounds, and game action so it can be communicated, changed, discussed, and better understood. This is done by artists who work very closely with the writers to insure that the nature of the original idea is not lost.


NAME OF GAME: Lazarus DATE: 1/1/2015

DESIGNERS YOUR NAME: Joy Stick PARTNER'S NAME: Johnny X. Box

NOTE: DO NOT PROCEED WITH THIS STORYBOARD UNLESS YOU HAVE COMPLETED A CONCEPT DOC.

Using your best sketching skills, draw images that represent what the scenes of your game will look like. Use some detail, but don't worry about perfection.

In the spaces below each scene, identify and list which items are sprites (controlled by events) and which items are objects. Briefly describe the mood, setting and backdrop. List your sounds and music ideas.

SPRITES: Stone box, cardboard box, wood box, Lazarus moving right
OBJECTS: Stone box, cardboard box, wood box, Lazarus moving right
MOOD, SETTING, AND BACKDROP:
Warehouse setting, playful, Lazarus has a playful smile as he leaps away from falling boxes.
SOUNDS/MUSIC: Background music, moving sounds like a slide

SPRITES: Stone box, cardboard box, wood box, Lazarus jumping right
OBJECTS: Stone box, cardboard box, wood box, Lazarus jumping right
MOOD, SETTING, AND BACKDROP:
Warehouse setting, more of a sense of urgency as boxes pile up. Lazarus jumps on boxes.
SOUNDS/MUSIC: Background music, jumping sounds like a spring

SPRITES: Stone box, cardboard box, wood box, Lazarus jumping right
OBJECTS: Stone box, cardboard box, wood box, Lazarus jumping right
MOOD, SETTING, AND BACKDROP:
warehouse setting, Lazarus has look of fear as boxes is about to crush him.
SOUNDS/MUSIC: Background music, oh no sound, squish sound when he is crushed

"<u>ELECTION DAY</u>"

<u>DESIGNERS</u>
Johnny X. Box, Suzy E. Rated, and Joy Stick
PERIOD 7

The object of this computer game is to become the mayor of the town. During a preset period of time, two players each control a sprite and move it through a series of streets and objects in the setting of a small town. The scene of the small town is represented on a street map. As the game is being played, patriotic music will be heard. Using keyboard events as controls, each player moves his sprite through the streets of the town which act as a maze in the game. The player must move his sprite through the maze and touch objects to score points, called votes. The game will open with a front end with celebratory music and an opportunity for players to input their names and register as a Democrat or a Republican.

The objects in the scene will represent homes, offices, businesses, stores, parks, etc. Each object will be programmed with controls that assign votes for only one player. When the player finds a vote, an applause sound will be heard and a vote for that player will be added to the score shown on a tally chart posted at City Hall. It will be up to the player to move his sprite as quickly as possible through the background image, which is a maze of streets and find the objects that have his votes.

The player may find that many of his votes are hidden in specific parts of the town. To get quickly from one part of the town to another, the player may find that some streets are more like highways that help him move faster. This all happens in a preset period of time called a voting day. The players can agree to adjust the length of a voting day by adjusting the time to be longer or shorter. The candidate who controls his sprite to gather the most votes by the end of the voting day will be the winner and be declared the mayor of the town. The sprite will then automatically change to include a mayor's badge and a top hat.

DO THIS ON YOUR TUTORIAL GUIDE

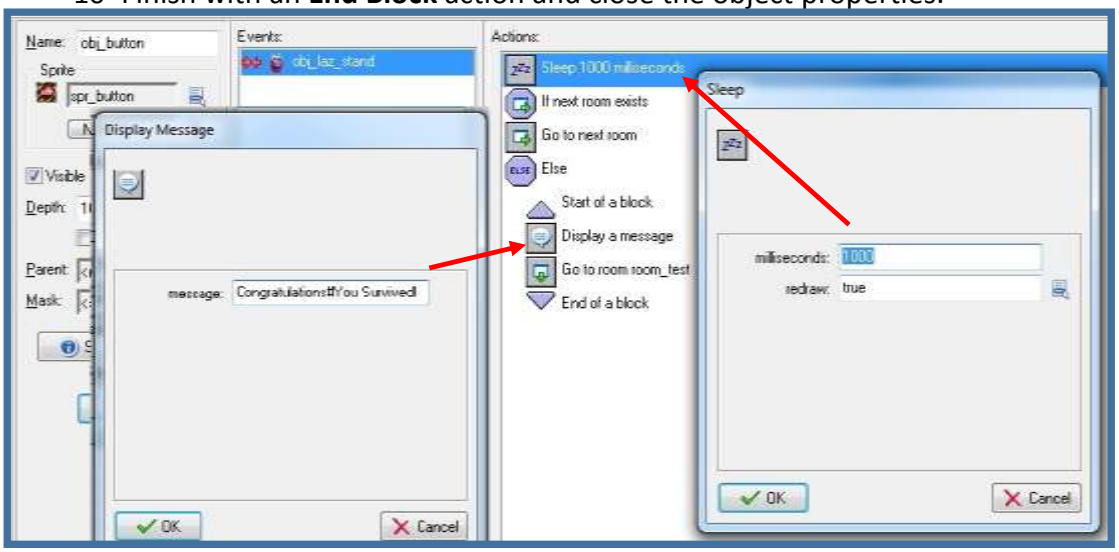
STAGE 5 HYPOTHESIS STATEMENTS: Now it's time to predict the behaviors of the Lazarus object and behaviors of the boxes, based on the check collision actions.

- **Describe the conditions that will cause the Lazarus object to change. What that change look like?**
- **What do you think Lazarus will look like? How will this make the animation *plausible*?**

STAGE 6:

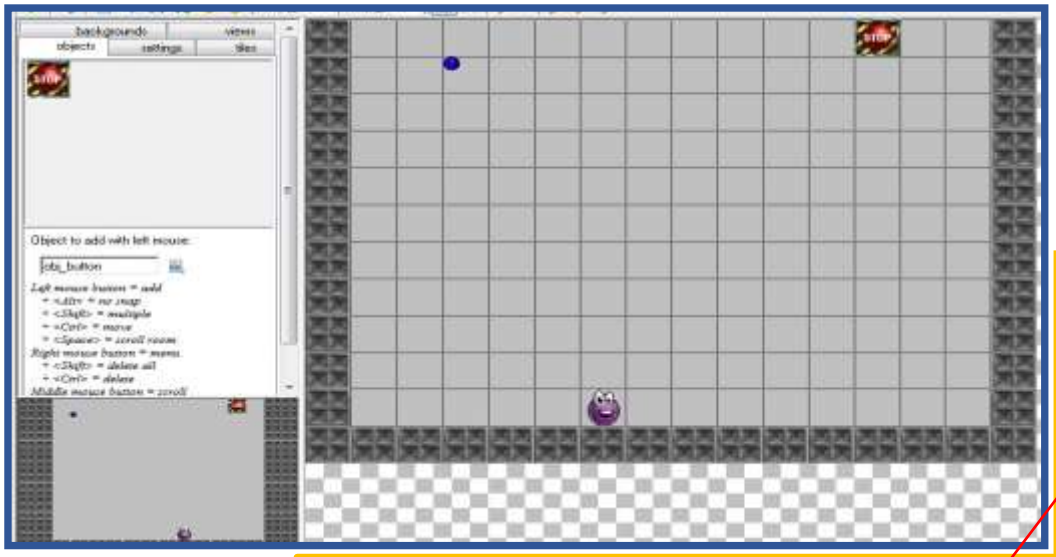
Creating a new button object resource for the game

1- Create a new sprite called ***spr_button*** using ***Button.gif*** (it looks like a STOP button).
2- Create a new object called ***obj_button*** and assign it ***spr_button***. Set **Depth** to **10**.
3- Add a **Collision** event with the ***obj_laz_stand***.
4- Drag and drop a **Sleep** action into the collision event from the **main2** tab. Set **Milliseconds** to **1000** (1 second) and **Redraw** to **true**.
5- Add a conditional **Check Next** action (**main1** tab).
6- Add a **Next Room** action (**main1** tab). Select a transition effect if you wish.
7- Add an **Else** action followed by a **Start Block** action.
8- Add a **Display Message** action (**main2** tab) and set **Message** to something like "**CONGRATULATIONS#You survirved!**". Do not use the quotation marks in the message.
9- Add a **Different Room** action and set **New Room** to the first room (***room_test***, which is the only room at the moment). Select a transition.
10- Finish with an **End Block** action and close the object properties.



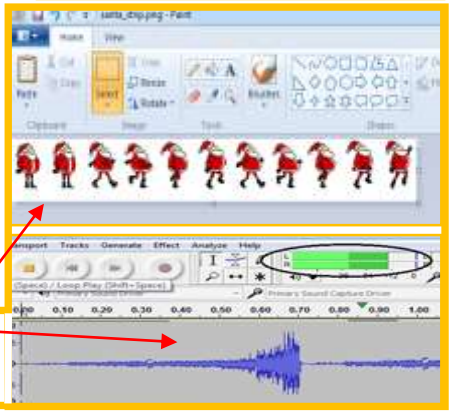
11- Open your test room and add a stop button at the top of the pit.

NOTE: You may adjust the location of this button later to make the game easier or more challenging. You can create easier game play by lowering the button or adding more buttons.



Game Development

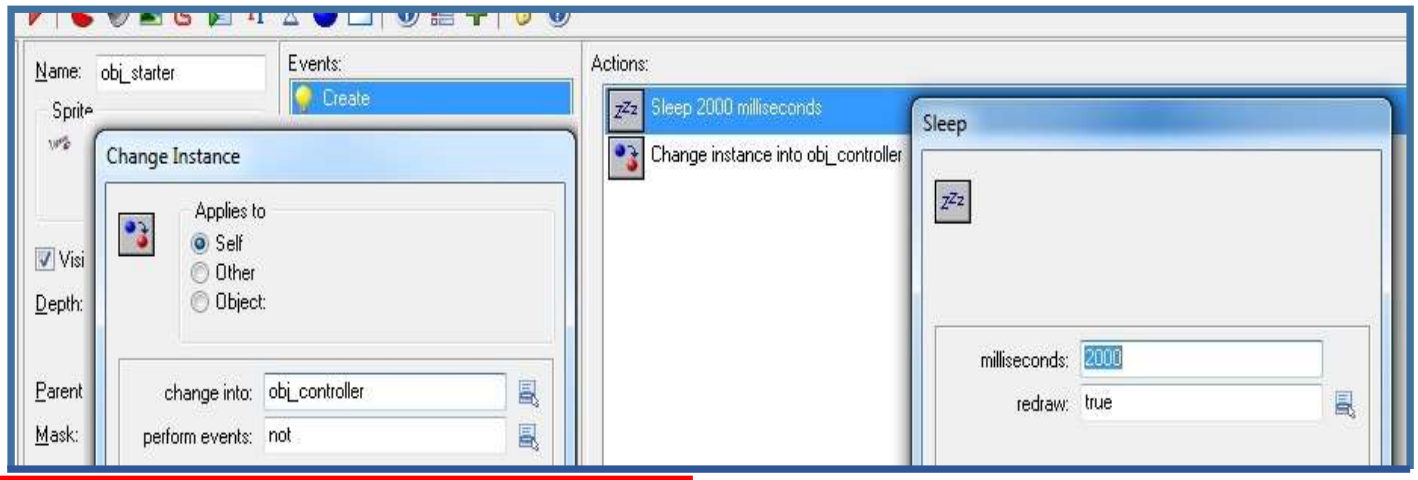
Once the concept phase is over, the technical side of game design begins. In **game development**, work that requires programming, scripting, and other technology skills is completed to build a playable game **prototype**. The concepts and story must be created in digital form. The art for objects, background, and animation must be created and formatted. Music must be written, and along with sound effects, it must be recorded and put into a digital format. Building these graphic and audio assets is called **asset development**. The *assets* are then applied later in the **prototyping** of the game. You will soon be challenged to develop your own original graphical and sound assets. You can use software like *Audacity*, *Gimp*, and *Paint* for basic **asset development**.



Simple graphic and sound **asset development** can be done on your computer using software like Paint or Audacity.

Creating a new starter object for the front end of the game

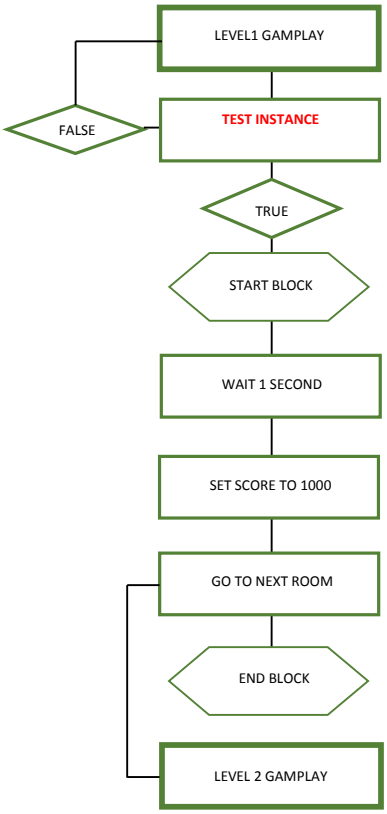
1- Create a new sprite called ***spr_title*** using ***Title.gif***.
2- Create a new object called ***obj_starter*** and give it the title sprite.
3- Add a **Create** event and include a **Sleep** action in it. Set **Milliseconds** to **2000**, for a wait of two seconds.
4- Include the **Change Instance** action and select the ***obj_controller***. Close the object properties.
5- Edit your test room, and remove the controller object using the right mouse button and selecting delete. Add the starter object at an appropriate place instead.



DO THIS ON YOUR TUTORIAL GUIDE

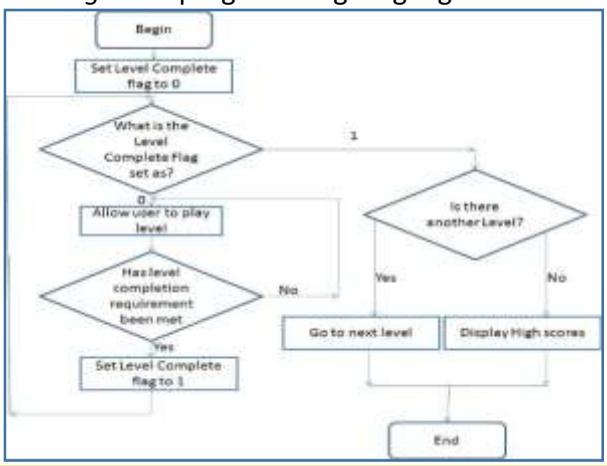
STAGE 6 HYPOTHESIS STATEMENTS: Now it's time to predict the behaviors of the button and title objects.

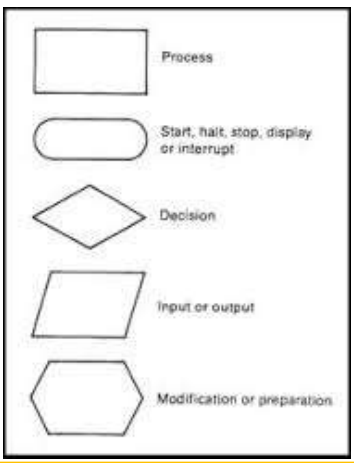
- **How will the game start at the *front end*?**
- **Is there a goal in the gameplay?**



More on Game Development: Making Flowcharts

Flowcharting is a process used by programmers to plan the sequence of programmed events and actions. Think about the creation of the logic systems that you programmed in your game builds. All of the cause and effect of events, conditional actions, and actions must be carefully thought out, and then mapped on a graphic organizer called a **flowchart**. A basic flowchart was used to illustrate a sequence of programmed events and actions in *Galactic Mail* (shown again on left). In both of the flowchart examples, the use of standard flowcharting symbols (see symbols) to show how programmed procedures, *sub-procedures* (also called *blocks* or *subroutines*) organizes the actions sequentially, so the game program can be **prototyped** using a *game engine* or programming language.





STAGE 7:

Sounds, Backgrounds, and Help (No illustrations)

1- Create sounds for ***Music.mp3***, ***Wall.wav***, ***Crush.wav***, ***Squished.wav***, ***Move.wav***, and ***Button.wav***. Use the usual naming conventions for sound resources (***snd_music***, ***snd_wall***, etc.).

2- Open the ***obj_controller*** and create a **Game Start** by clicking **Add Event** and then clicking **Other** to find it in the drop-down menu. Drag and drop a **Play Sound** action, select you **Music** sound, setting **Loop** to **true**.

3- Open each of the falling box objects one at a time and perform the following:

- Select the collision action with the ***obj_wall***, add a **Play Sound** and select ***snd_wall*** leaving Loop as false.
- For each collision event with a box that it should crush, add a **Play Sound** and select ***snd_crush***. For example, since a falling stone object crushes every box but itself, add this action to the collision event with all other boxes (except for the stone itself). A falling cardboard box won't crush anything, so it will not need any crush sounds with the collisions.

4- Open ***obj_laz_squished*** and add a **Create** event with a **Play Sound** action of ***snd_squished***. This will cover all squishing collisions.

5- In all four of the moving Lazarus object (left, right, jumping left, and jumping right), add **Create** events with **Play Sound** actions of ***snd_move***.

6- Open the ***obj_button***, and add a **Play Sound** action of ***snd_button*** to the existing collision with ***obj_laz_stand***.

Adding a background to the room

1- Create a background using ***Background.bmp*** and give it a typical name following naming conventions.

2- Reopen the room, select the backgrounds tab, and add the background from the menu list (found under the Foreground image property selector).

Finishing Touches

Decide how difficult you want your game to be? You can make it easier by lowering the button, adding additional buttons, adding wall blocks configured as stairs, or all of these. You can make the game harder by leaving one row of wall blocks along the bottom, left, and right sides of the room. In this configuration, you would want the button to be as high as possible, close to either the right or left side of the room.

Game Information

Open **Game Information** in the resource list. Use the same basic scheme shown in the *Galactic Mail* tutorial, including event keys and credits. The names for the game credits are the same.

DO THIS ON YOUR TUTORIAL GUIDE

STAGE 7 CONDITIONAL STATEMENTS: Write conditional IF/THEN statements for steps 2 through 6 in STAGE 7. For step 4, just choose any one the Lazarus objects.

Now it's time to test your game, so go ahead and click on the green triangle (▶) on the menu bar to run the game normally.

DOES THE ACTION THAT YOU SEE AND THE CONTROL OF THE OBJECTS MEET YOUR HYPOTHESIS STATEMENTS?

The basic gameplay of Lazarus is still as it was at the end of Stage 4, except now there is a goal for the player to achieve. As the boxes (the ones that don't get crushed) continue to pile up, Lazarus should be able to jump onto no more than one box on the left or right. Due to the conditional actions you created for collision checks, when Lazarus is in between two stacks of boxes, he will show a look of fear. The animation technique of *anticipation* is evident here, as it tells the player that Lazarus is about to be crushed. And of course, the animation continues as Lazarus gets crushed. Although all of this is somewhat exaggerated for effect, the behaviors of Lazarus reflect an emotional reaction that we can all understand, making his ending *plausible*. The gameplay ends here with a little taunting in the message, including a message encouraging the player to continue. The game will restart so the player can keep practicing to reach the goal of pressing the button. If that happens, the player gets a congratulatory message to celebrate the success of reaching the goal, and of course, Lazarus' survival.

DO THIS ON YOUR TUTORIAL GUIDE

STAGES 5, 6, & 7 TEST AND EVALUATION: Write an evaluation of your hypothesis and programming properties from STAGE 4.

- **If you state "valid", provide a detailed explanation of "why" the object behaves properly based on your hypothesis, and the events and actions that you programmed.**
- **If you state "invalid", make sure that you expose your errors in reasoning, or your errors in programming.**

Additional challenges

Before experimenting any further, make sure you go to **File** and **Save as** and rename several copies of your game build file (*initials_lazarus1*, *initials_lazarus2*, etc.). Use these files in case you make fatal errors and need to restart. At least, you will always have your working copy of Lazarus. Then, with the new game build files, try the following:

- Add additional rooms with harder gameplay. You can make copies of the existing room and make changes to add difficulty, as explained near the end of the tutorial. You would need a Next Room action to occur along with the collision of boxes and Lazarus, with a short pause in between levels.
- Editing the controller object to add cheats. These will help you navigate from room to room while you are attempting these challenges. They can be removed when you are done if you don't want them in your final game.
  1- Open the properties form for the controller object.
  2- Add a Key Press, <N> event and include the Next Room action.
  3- Add a Key Press, <P> event and include the Previous Room action.

TUTORIAL CONTINUES ON NEXT PAGE

Game Development: The Prototyping of the Game

When you play a game in an arcade, on a game console in your home, or just sitting at your computer, you are interacting with a product that was the result of a creative process. The final phase of that process is the coding, or programming of the game to make a test version, called a **prototype**. The coding, scripting, or programming can be done using programming languages like C, C+, C# or Java. Sometimes game building engines, including advanced versions of *Game Maker*, or Unity are used in **prototyping**. The result is a working game that can be played and tested. Although it is not ready to be seen by the public, the **prototype** version of the game can be tested for gameplay and playability. Problems, or errors, in the program can be detected during this process. Programmers and game testers call these **bugs**. The problems are studied, and solutions to those problems are created and applied to the correct the prototype. This process of searching out and correcting errors is called **troubleshooting**, or **debugging**. Once it is decided that the prototype is playable outside of the development process, testing can be done by people knowledgeable in computer/video games. Testing in the game design industry includes a first test, called *alpha testing*. This is usually by testers working for the game company. Further improvements, or changes to the game can be made before people outside of the game company see it. It is then turned over to a select group of people outside of the company in a second round of testing, called *beta testing*. The *beta testers* will report back to the company, often through electronic surveys, email, and even phone conversation. The information that they return to the company may be used to further develop the game before the final version is packaged and sold.

As with some of the other concepts you studied in this unit, you will soon be required to develop an original game. That game will be *alpha* and *beta* tested. Game testing vocabulary and concepts will be covered deeper at that time.

DO THIS ON YOUR TUTORIAL GUIDE

STAGES 5, 6 & 7 CONCEPT SUMMARY: **Review the concepts found in the gold boxes throughout the tutorial. Write a full paragraph describing the processes of concept development and game development. Be sure to correctly use vocabulary from these stages in your writing.**

END OF STAGES 5, 6, & 7. REVIEW YOUR WORK ON THE TUTORIAL GUIDE. STUDY THE TUTORIAL GUIDE RIGOR SCALE. MAKE REVISIONS AND IMPROVEMENTS BASED ON THE SCALE. UPLOAD COMPLETED TUTORIAL GUIDES TO EDMODO.