Checklist of Team 12: Termanator
Donald Eng, Rodrigo Ipince, Kevin Luu

---

*Note: basic features are marked by numbers. More advanced features are marked by *, which means that these goals will improve the game, but not in an absolutely necessary manner.*

1. Devices

1.1. Shaker
- 1.1.1. Physical Implementation of Shaker (constructed by Donald)
- 1.1.2. Converting shaker's level signals into pulses (written by Kevin)
- 1.1.3. Converting time difference between consecutive pulses into predefined power level (written by Kevin)
- 1.1.4. Implement moving average filter to accompany extraneous pulses (written by Kevin)
- 1.1.5. Display of Power Sprite Bar on XVGA, corresponding to shaker's intensity (written by Kevin)

2. Inputs

2.1. Video Input
- 2.1.1. Modification of NTSC-to-ZBT module to down sample 24-bit color (8-bits each) to 18-bit color (6-bits each) (Modification to Full Motion Dance Machine's module by Kevin)

2.2. Video Processing
- 2.2.1. Implementation of Color Space Converter, from YCrCb to RGB (written by Kevin)
- 2.2.2. Implementation of Color Space Converter, from RGB to HSV* (written by Kevin)
- 2.2.3. Implementation of frame filter
  - 2.2.3.1. Filter pixels that falls within the set threshold (written by Kevin)
  - 2.2.3.2. Determine the frame that encloses the detected marker (written by Kevin)
- 2.2.4. Implementation of pointer calculation
  - 2.2.4.1. Calculate ongoing average of x and y position to determine center of mass for detected marker (written by Kevin)

3. Game Play

3.1. Creature motion
- 3.1.1. Random number generator to choose new paths for creatures (written by Rodrigo)
- 3.1.2. Show creatures moving down paths and reappearing on new ones (written by Rodrigo)
- 3.1.3. Change creature speed as frame rate varies (used to increase game's difficulty) (written by Rodrigo)
- 3.1.4. Have the number of active creatures at any point in time be variable* (written by Rodrigo)

3.2. Projectile motion
- 3.2.1. Show a few moving projectiles and how their trajectory and speed depend on the position of the pointer and the power, respectively (written by Rodrigo)
- 3.2.2. Have projectiles correctly determine whether they collide of not with each creature (written by Rodrigo)

3.3. Game logic
- 3.3.1. Increase frame rate as time progresses (written by Rodrigo)
- 3.3.2. Show collision logic that determines whether a collision is valid or not (written by Rodrigo)
- 3.3.3. Logic for starting, pausing, and ending the game (written by Rodrigo)

4. Graphics

4.1. 3D to 2D transforms
- 4.1.1. Sprite Scaling: the sprites should be scaled in size depending on their Z locations (smaller farther away, larger close up) (written by Donald)

4.1.2. Sprite Positioning: the sprites should be located according to a perspective depending on their Z locations (scaling X Y and normalizing) (written by Donald)

4.2. Sprite Drawing

4.2.1. Display Sprites from RAM/ROM: an original *.bmp should be displayed on the XVGA through the RAM/ROM through some file parsing

4.2.2. Animate Sprites Walking: show a basic loop animation based on Z for a sprite walking down a path (written by Donald)

4.3. Sprite Pipelining

4.3.1. Screen shifting: pipelining the circuit should screen shift pixels, show the shifting and the corrective shifting (written by Donald)

5. Miscellaneous

5.1. Background sound and sound effects (written by All)