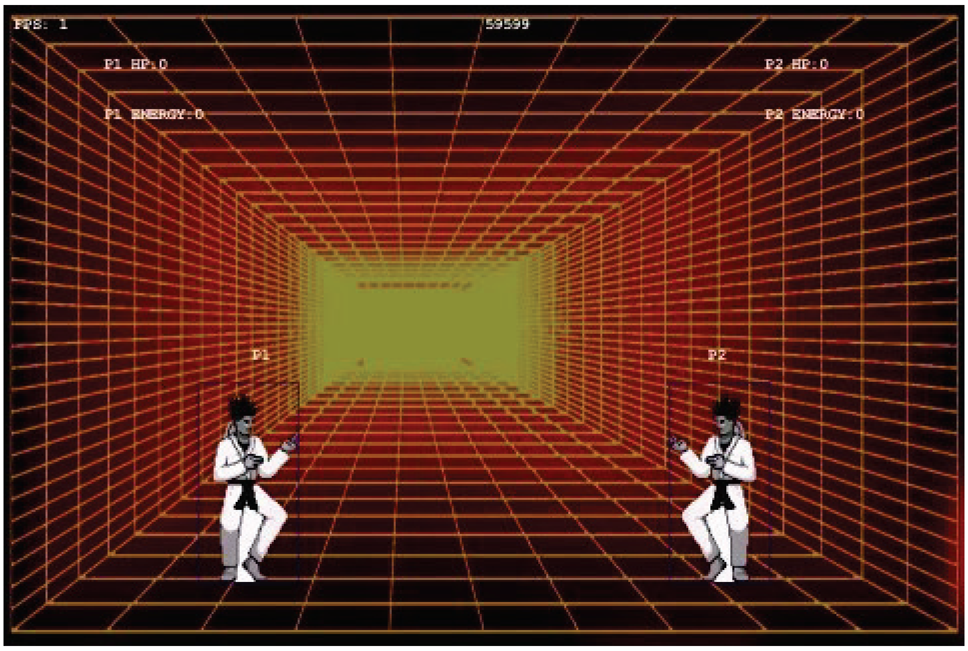
Introduction of the game-processing flow in a round

Initialization (the following settings are done by the system)

1. Clear the object data of the last round.
2. Set both players' hp, energy, and speed to 0.
3. Set P1's coordinates to (125,320), the left-hand side, and P2's coordinates to (580,320), the right-hand side.
4. Set both players' state to "STAND", action to "NEUTRAL".
5. Set the time limit to 60 seconds.



The following steps will be repeated until the round is over.

Step.1 Get the input information from AI(s) or human player(s) via keyboard.

Step.2 Update both characters' parameters.

A) Add SpeedX's value to the character's x-coordinate and SpeedY's value to the character's y-coordinate.

B) Decrease 1 unit from SpeedX to simulate the friction if the character's ~~x~~-coordinate = 320 (on the ground) and 1 unit from SpeedY to simulate the gravity if the character's y-coordinate < 320 (in the air).

C) Decrease 1 frame from remainingFrame.

D) If the character's y-coordinate > 320:

I. If LandingFlag of the current action is true, then set SpeedY to 0 and execute the landing action.

II. Set y-coordinate to 320.

E) If the current frame is the first active frame, generate the ~~attack~~ object.

F) If the remainingFrame is 0, set the next action to the one corresponding to the current condition as shown in the following table. In this table, the actions are listed in decreasing priority from top to bottom. The highest-priority action whose condition 1 or 2 holds will be used.

action	condition 1	condition 2
DOWN	action=CHANGE_DOWN	
RISE	action=DOWN	
AIR	state=AIR	y-coodinate<320
STAND	Default	

G) If the characters overlap, compare their speedX's absolute value and horizontally push back the slower one with the speed difference.

H) If the left boundary of the character's hit box is lower than 0 or the right boundary is larger than maxStageX, set the character's hit box's left boundary to 0 or right boundary to maxStageX. Moreover, if the state is down, first halve speedX's absolute value and then reverse the sign.

Step.3 Execute the action according to the key input(s) and the current state. If the action's isControl is true, the action will be executed. Otherwise, the action will not be executed. For executing an action,

A) Update character.action to that action.

B) Based on the data on motion.csv, update character.state, speedX, speedY, hit, attack, control, and remainingFrame

Step.4 Move and remove the attack hit box.

A) The character and its attack hit box will move by a distance based on Attack's speedX and speedY.

B) Compare nowFrame and Active, if nowFlame is larger, the attack hit box will be removed.

Step.5 Check if the character's attack hit box and the opponent's hit box overlaps or vice versa; if this is true then perform the following steps for both characters:

A) If the block is succeeded, execute ***_GUARD_RECOV. Otherwise execute ***_RECOV. The following table shows different types of combinations based on various Attack types and Guard actions.

Guard action / Attack type	STAND_GUARD	CROUCH_GUARD	AIR_GUARD
High	block	block	block
Middle	block	hit	block
Low	hit	block	hit
Throw	hit	hit	miss

B) If attackType of the attack hit box is Throw, ~~the character~~ execute THROW_HIT ~~and the opponent character~~ execute THROW_SUFFER~~;~~

C) Calculate the damage.

D) Calculate the energy.

E) Change speedX and speedY according to impactX and impactY if the attack is not blocked.

F) If downProperty is true and the attack is unblocked,
execute CHANGE_DOWN.

G) Clear the hit box.

Step.6 Update FrameData.