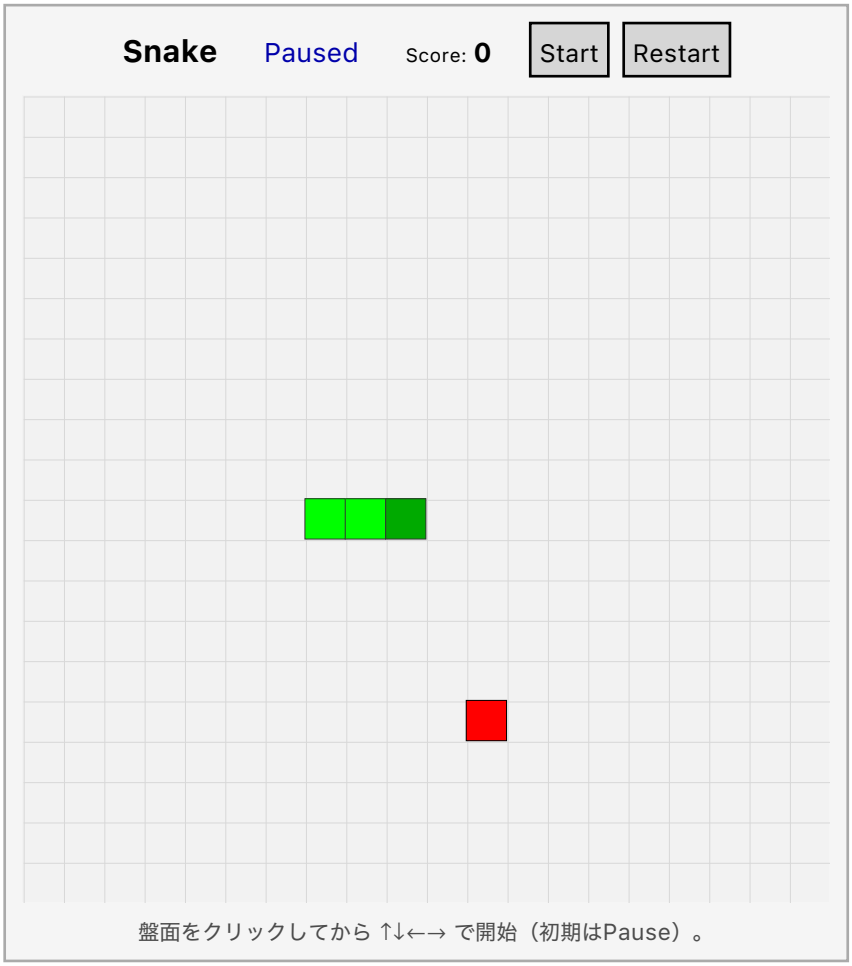
snakeゲーム

ChatGPT5君の提示：一発目からエラーはなし。ただし挙動はおかしい。

1. ShitReturn直後、snakeが自動的に右に移動し、右端で停止
2. （その後、何度か修正後）矢印キークリック直後、自動で端まで移動
3. （その後、何度か修正後）今度は矢印キークリックに反応せず

In[ ]:=
```
(*---Snake (paused start/arrow to start)---*)
DynamicModule[{w = 20, h = 20, snake, dir, food, alive,
  speed, score, paused, init, step, randomFood, setDir, draw},
 randomFood[] := Module[{p}, p = {RandomInteger[{1, w}], RandomInteger[{1, h}]};
   While[MemberQ[snake, p], p = {RandomInteger[{1, w}], RandomInteger[{1, h}]}];
   p];
 init[] := (snake = {{Ceiling[w / 2], Ceiling[h / 2]},
     {Ceiling[w / 2] - 1, Ceiling[h / 2]}, {Ceiling[w / 2] - 2, Ceiling[h / 2]}};
   dir = {1, 0};
   (*既定方向は右だが...*)paused = True;
   (* ←初期は停止*)food = randomFood[];
   alive = True;
   speed = .12; (*秒/ステップ*)score = 0;);
 step[] := Module[{next}, If[! alive || paused, Return[]];
   (* ←停止中は進めない*)next = snake[[1]] + dir;
   If[next[[1]] < 1 || next[[1]] > w || next[[2]] < 1 || next[[2]] > h, alive = False;
    Return[];];
   If[MemberQ[Rest[snake], next], alive = False; Return[];];
   If[next === food, snake = Prepend[snake, next];
    score++;
    food = randomFood[], snake = Prepend[Most[snake], next]];];
 setDir[new_] := If[alive && new =!= -dir, dir = new;
   paused = False];
 (* ←矢印入力で解除*)draw[] := Graphics[{{GrayLevel[.95], Rectangle[{0, 0},
     {w, h}]}, Table[{GrayLevel[.85], Line[{{i, 0}, {i, h}}]}, {i, 0, w}], Table[
    {GrayLevel[.85], Line[{{0, j}, {w, j}}]}, {j, 0, h}], {Red, EdgeForm[Black],
    Rectangle[food - 1, food]}, {Darker[Green], EdgeForm[GrayLevel[.2]],
    Rectangle[snake[[1]] - 1, snake[[1]]]}, {Green, EdgeForm[GrayLevel[.2]],
    Map[Rectangle[# - 1, #] &, Rest[snake]]}}, PlotRange → {{0, w}, {0, h}},
   ImageSize → 420, Axes → False, Background → White];
 init[];
 Panel@Column[{Row[{Style["Snake", 16, Bold], Spacer[20],
     Dynamic@If[alive, If[paused, Style[" Paused ", Darker@Blue, 14],
        Style[" Playing ", Darker@Green, 14]], Style[" Game Over ", Red, 14]],
     Spacer[20], Dynamic@Row[{"Score: ", Style[score, 14, Bold]}],
     Spacer[20], Button[Dynamic[If[paused, "Start", "Pause"]],
      paused = Not@paused, Method → "Queued"],
     Spacer[8], Button["Restart", init[], Method → "Queued"]}],
   EventHandler[Dynamic[If[alive, step[]];
     draw[], UpdateInterval → Dynamic[speed], TrackedSymbols :> {}],
    {"UpArrowKeyDown" :> setDir[{0, 1}], "DownArrowKeyDown" :> setDir[{0, -1}],
     "LeftArrowKeyDown" :> setDir[{-1, 0}],
     "RightArrowKeyDown" :> setDir[{1, 0}]}, PassEventsDown → True],
   Style["盤面をクリックしてから ↑↓←→ で開始（初期はPause）。", 12, GrayLevel[.3]]},
  Spacings → 1, Alignment → Center]]
```

Out[ ]=



イベントハンドラーの設定・挙動を疑う

1. 同じイベントでもマウスはclickされたかどうかの判断なので楽
2. これに対し特定のキーが押されたかどうかの判別は難しい
3. それでも最初はChatGPT5にお任せで修正していた。
4. 後半はこちらが手動しつつ、でも正解はChatGPT5が出す例が**ヒント**になった。
5. AIにデバッグコードを生成させ、一緒にデバッグする！効果あり。
6. 状況によってはシンプルな版をベースに再構築する。
7. トップダウンで問題なければOK。問題あればボトムアップに構築。

例１：mouse click

In[ ]:=
```
DynamicModule[{col = Green},
 EventHandler[Style["テキスト", FontColor → Dynamic[col]],
  {"MouseClicked" :> (col = col /. {Red → Green, Green → Red})}]]
```

Out[ ]=

テキスト

例２：矢印キー click : 反応なし

In[•]:=
```
DynamicModule[{x = "key"},
 EventHandler[x, {"KeyDown" :> (x = CurrentValue["EventKey"])}]]
```

Out[•]=

key

例３：反応あり、ただし字化け

In[•]:=
```
DynamicModule[{x = "key"}, EventHandler[Dynamic@Style[x, 18],
  {"KeyDown" :> (x = CurrentValue["EventKey"])}, Method → "Preemptive",
  (* ←すぐ処理*)PassEventsDown → False  (* ←他に流さない*)]]
```

Out[•]=

?

例４：文字コードに対応、ただし矢印キーにヒットせず

In[•]:=
```
DynamicModule[{x = "key"}, EventHandler[Dynamic@Style[x, 18],
  {"KeyDown" :> Module[{k = CurrentValue["EventKey"], c}, c = ToCharacterCode[k];
     (*例: ←8592, ↑8593, →8594, ↓8595*)
     Which[c === {8592}, x = "Left", c === {8593}, x = "Up", c === {8594}, x = "Right",
      c === {8595}, x = "Down", True, x = k   (*それ以外のキーは生値*)]],
  Method → "Preemptive", PassEventsDown → False]]
```

Out[•]=

?

例５：MacOS固有の文字コードへの対応

In[•]:=
```
arrowMap = <|{63 234} → "Left",
    {63 235} → "Right", {63 232} → "Up", {63 233} → "Down"|>;
EventHandler[Dynamic@Style["(press arrow keys)", 16],
 "KeyDown" :> Module[{c = ToCharacterCode[CurrentValue["EventKey"]]},
   If[
    KeyExistsQ[arrowMap, c], Print[arrowMap[c]],
    Print[FromCharacterCode[c]]]]]
```

Out[•]=

(press arrow keys)

```
d
d
f
r
t
Up
Left
Down
Right
```

例６：dynamic変数 x を用意し、その値を書き換える

```
In[•]:= arrowMap = <|{63 234} → "Left",
    {63 235} → "Right", {63 232} → "Up", {63 233} → "Down"|>;
DynamicModule[{x = "key"},
 EventHandler[Dynamic@Style[x, 16],
  "KeyDown" :↦ Module[{c = ToCharacterCode[CurrentValue["EventKey"]]},
    If[
     KeyExistsQ[arrowMap, c], x = arrowMap[c],
     x = FromCharacterCode[c]]]]]
Out[•]=
Right
```

例７：xをグラフィックスとして矢印キーで移動

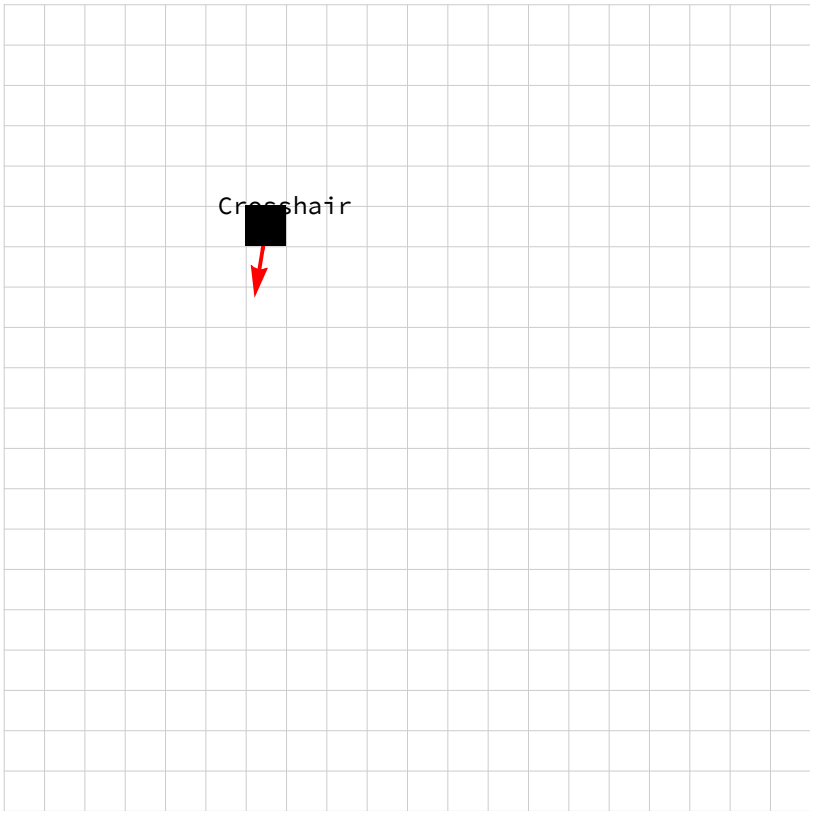
見事に失敗。グラフィックス上に矢印キーイベントをハンドリングする機能なし！

例８：［暫時完成版］LocatorPane

ChatGPT5の大きなミス：Clipの引数間違え！

In[ ]:=
```
DynamicModule[{w = 20, h = 20, pos = {10, 10}, loc,
  thr = 0.35, delta = {0., 0.}, dist = 0., movedQ = False}, loc = pos;
 (*初期ロケータは石の位置*)Row[{LocatorPane[Dynamic[loc, (loc = #;
      delta = N[loc - pos];
      dist = Norm[delta];
      movedQ = dist ≥ thr;
      If[movedQ, Module[{step}, step = If[Abs[delta[[1]]] ≥ Abs[delta[[2]]],
          If[delta[[1]] > 0, {1, 0}, {-1, 0}], If[delta[[2]] > 0, {0, 1}, {0, -1}]];
        (** ここを修正：各成分ごとに Clip*)
        pos = {Clip[pos[[1]] + step[[1]], {1, w}], Clip[pos[[2]] + step[[2]], {1, h}]};
        loc = pos;
        (*ロケータは石の上に戻す*)FinishDynamic[];   (*直後に反映*)];]) &],
  Dynamic[Graphics[{(*格子線*)Table[{GrayLevel[.8], Line[{{i, 0}, {i, h}}]},
     {i, 0, w}], Table[{GrayLevel[.8], Line[{{0, j}, {w, j}}]}, {j, 0, h}],
    (*デバッグ矢印：石中心→ロケータ*){Directive[Red, Thick], Arrow[{pos - .5,
      pos - .5 + delta}]}, (*石（1×1）*){Black, Rectangle[pos - 1, pos]}},
   PlotRange → {{0, w}, {0, h}}, Axes → False, ImageSize → 420],
  TrackedSymbols :↦ {pos, loc, delta} (*追跡はこの3つで十分*)],
 LocatorAutoCreate → False, Appearance → "Crosshair"], Spacer[16],
(*独立ログで pos が更新しているか確認*)Dynamic[Column[{Style["DEBUG", Bold],
   "pos   = " <> ToString[pos], "loc   = " <> ToString[loc],
   "delta = " <> ToString[NumberForm[delta, {3, 2}]], "dist  = " <>
    ToString[NumberForm[dist, {3, 2}]], "moved?= " <> ToString[movedQ]},
  Spacings → .25], TrackedSymbols :↦ {pos, loc, delta, dist, movedQ}]}]]
```

Out[ ]=



DEBUG
pos = {7, 15}
loc = {7, 15}
delta = {-0.29, -1.78}
dist = 1.80
moved?= True