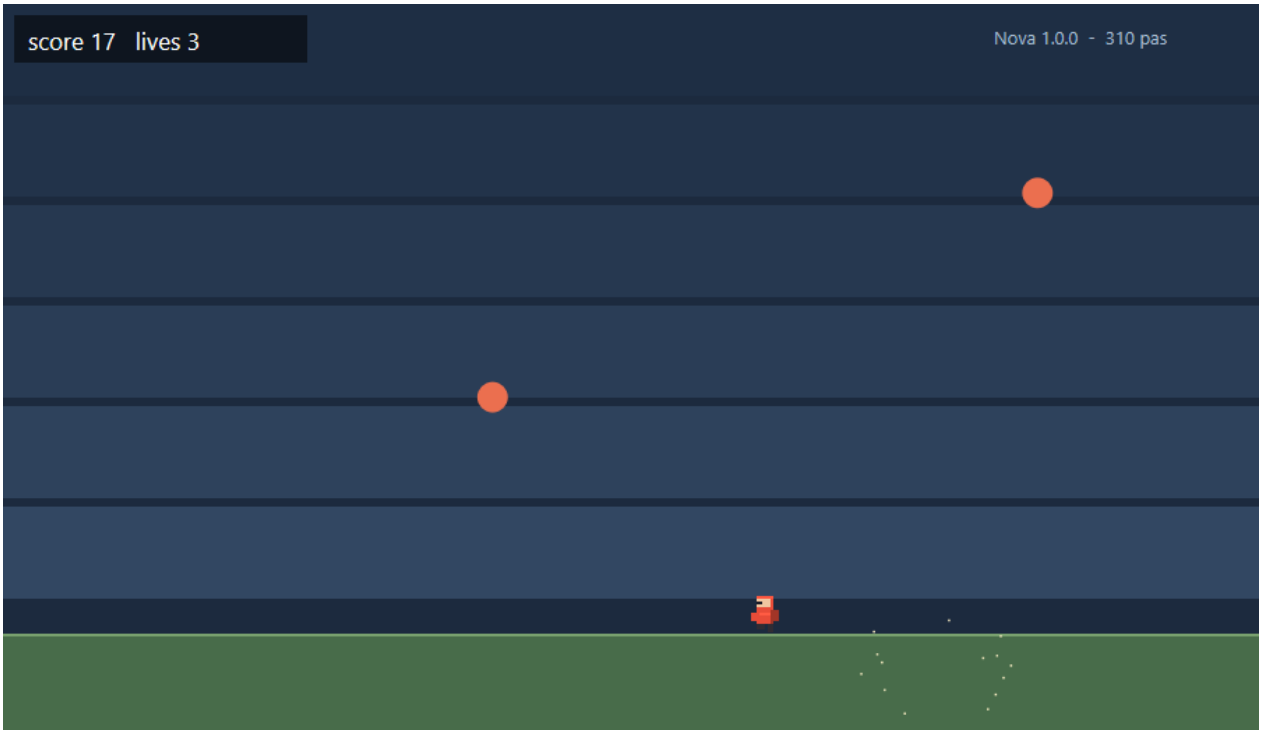
Nova

Make Your First 2D Game



The game this course builds, one frame before the catch. Chapter 12 is the whole listing of the file that painted it.

What you will build, in nine steps

A window that opens • a character that walks • fruit that falls • a score and three lives • a title screen and a game over • particles and sound • a game that tests itself.
Every step is a program you can run and that keeps working in the next one. Nothing is assumed but AutoIt basics: variables, `If`, `Func`.

Version 1.0 • AutoIt 3.3.16.1, 32-bit and 64-bit • Direct2D + DXGI + DirectWrite
MIT licence • Numeric1 • the engine is four .au3 files and one door: Nova.au3

Contents

1. Before you type a single line 3
2. A game is a loop, and that is the whole idea 3
3. Step 1 - a window that opens and closes 4
4. Step 2 - put the character on screen 4
5. Step 3 - make it walk 5
6. Step 4 - make something fall 6
7. Step 5 - catch it, and count it 7
8. Step 6 - losing, and the two numbers that change the game 8
9. Step 7 - three screens, one number 9
10. Step 8 - juice: particles and sound 9
11. Step 9 - a game that tests itself 10
12. The finished program 11
13. Make it yours 21
14. When it does not work 21
15. Words this course uses 22
16. Where to read next 22

A course, not a reference. You start with an empty file and you finish with a game that has a title screen, a score, three lives, particles, sound and a self-test. Each step is a program you can run: type it, run it, look at the window, then read why it works. Everything is built on the same nine ideas, so nothing is thrown away between steps.

What you need: a Windows PC, AutoIt 3.3.16.1 (32-bit or 64-bit, both work), and the Nova package you unzipped. No compiler, no install step, no other library.

01 Before you type a single line

Where to put your file

Unzip `Nova-1.0.zip`. You get one folder:

```
Nova-1.0/
    Nova.au3          <- the door you open the engine with
    LICENSE           <- MIT, you may do almost anything
    MANIFEST.md       <- what every file is for
    docs/             <- this course, the quick start, the API
    examples/         <- 01_hello.au3, 02_plateforme.au3, 03_catch.au3
        assets/       <- player.png, tiles.png, coin.wav, mort.wav
    src/              <- the engine itself, you never edit this
    tests/  tools/    <- the proofs, only for the suite
```

Save your own program **inside** `examples`, next to the other examples. That is not a style rule: it is what makes the two paths in your code work, because both are relative to your file.

```
#include "../Nova.au3"                          ; the engine is one folder up
$g_hPerso = _Nova_LoadImage(@ScriptDir &; "\assets\player.png")   ; art is in assets
```

How to run a script

You do	Result
Press F5 in SciTE	runs the file you are editing, console at the bottom
Drag `mon_jeu.au3` onto `AutoIt3.exe`	runs it, no console
Type `AutoIt3.exe mon_jeu.au3` in a terminal	runs it, you choose 32 or 64-bit

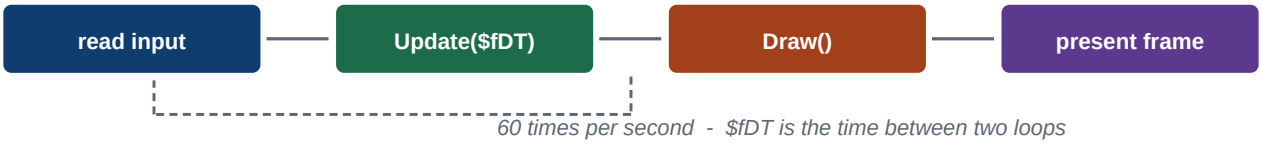
Use the same interpreter you plan to ship. If you ship 64-bit, test 64-bit.

> **Tip** if nothing happens when you run a script, run it from a terminal with `AutoIt3.exe /ErrorStdOut mon_jeu.au3`. AutoIt then prints the error instead of showing a dialog you have to click.

What you are about to build

A character walks along the bottom of the screen. Fruit falls from the top. Walk under it to catch it: +1 point. Let it hit the floor: you lose one of three lives. Every six falls there is a gold star worth 5 points that drops faster. The game speeds up as your score grows, and a title screen and a game-over screen wrap it all. That program is `examples/03_catch.au3`, and the last chapter prints it whole.

02 A game is a loop, and that is the whole idea



A 2D game does not do a sequence of things. It does the same small thing over and over, 60 times a second:

```
read the keyboard and the mouse  ->  Update($fDT)  ->  Draw()  ->  show the frame
                        ^                                          |
                        +------------------------------------------+
```

`Update` changes the numbers that describe your world: where the character is, how fast the fruit falls, how many lives are left. `Draw` reads those numbers and paints a picture. Neither one knows about the other. Once you feel that split, the entire API is obvious, because every function is either an input, a number, or a paint.

Nova gives you the loop. You give it two function names.

03 Step 1 - a window that opens and closes

Save this as `mon_jeu.au3` in `examples` and run it.

```
#include "../Nova.au3"

Func Update($fDT)
    If _Nova_KeyHit($NOVA_VK_ESCAPE) Then _Nova_Quit()
EndFunc

Func Draw()
    _Nova_ClearColor(_Nova_RGB(28, 42, 62))
EndFunc

If Not _Nova_Create("My game", 900, 520) Then
    Exit MsgBox(48, "Nova", _Nova_LastError())
EndIf

_Nova_Run("Update", "Draw", 60)
_Nova_Destroy()
```

You should see a dark blue window, and pressing Escape closes it. That is a real game loop: `Update` runs 60 times a second, `Draw` runs once per frame.

Read it from the bottom up, which is how AutoIt runs it:

Line	What it does
`_Nova_Create("My game", 900, 520)`	makes the window and the whole graphics chain (Direct2D, DXGI, DirectWrite). Returns `0` on failure.
`_Nova_Run("Update", "Draw", 60)`	starts the loop and **blocks** until the window closes. `60` is the logic rate.
`_Nova_Quit()`	asked from inside the loop: stop, so `_Nova_Run` returns.
`_Nova_Destroy()`	releases everything, in the reverse order it was created.

Trap

every engine function that paints returns `0` with `@error = $NOVA_ERR_STATE` when you call it outside `Draw()`. That is not pedantry: a paint outside the frame would corrupt it. `_Nova_Capture()` is the one exception you must remember - it belongs in `Update`, never in `Draw`.

04 Step 2 - put the character on screen

One picture file, one handle, one call. A handle is a plain number: `0` always means failure, in every function in this engine.

```
#include "../Nova.au3"

Global $g_hPerso = 0
Global $g_X = 434, $g_Y = 420

Func Update($fDT)
    If _Nova_KeyHit($NOVA_VK_ESCAPE) Then _Nova_Quit()
EndFunc
```

```
Func Draw()
    _Nova_ClearColor(_Nova_RGB(28, 42, 62))
    _Nova_FillRect(0, 452, 900, 68, _Nova_RGB(72, 108, 74))
    _Nova_Image($g_hPerso, $g_X, $g_Y)
EndFunc

If Not _Nova_Create("My game", 900, 520) Then
    Exit MsgBox(48, "Nova", _Nova_LastError())
EndIf

$g_hPerso = _Nova_LoadImage(@ScriptDir &; "\assets\player.png")
If $g_hPerso = 0 Then
    MsgBox(48, "Nova", _Nova_LastError())
    _Nova_Destroy()
    Exit
EndIf

_Nova_Run("Update", "Draw", 60)
_Nova_ImageFree($g_hPerso)
_Nova_Destroy()
```

A red figure now stands on a green strip. Two things to notice:

- `(x, y)` of a sprite is its **top-left corner**, in pixels, measured from the

top-left of the window. `452 + 68 = 520`, so the floor fills the bottom strip.

- `player.png` is 128 pixels wide and 32 tall: it is four 32x32 frames in a row.

`_Nova_Image` paints the whole strip. To take one frame you use `_Nova_ImagePart`, and to know which frame you use `_Nova_AnimFrame`, which turns a time into a frame number.

```
; the frame that matches this instant, out of 4 frames, played at 8 per second
Local $iFrame = _Nova_AnimFrame(_Nova_Time(), 4, 8)
_Nova_ImagePart($g_hPerso, $iFrame * 32, 0, 32, 32, $g_X, $g_Y)
;                  source x   source y  w   h   where
```

`_Nova_ImagePart` accepts four optional arguments after the position, so the full shape you will meet in `03_catch.au3` is:

```
_Nova_ImagePart($hImg, $sx, $sy, $sw, $sh, $x, $y, $w, $h, $fOpacity, $bFlipX, $bFlipY, $fAngle)
```

Leaving `$w` and `$h` at `-1` means "draw the piece at its native size". The last three are the fun ones: opacity `0.0..1.0`, mirror, and rotation in degrees around the centre of the sprite.

05 Step 3 - make it walk

Now the first number that matters, `$fDT`. It is the time in seconds between two `Update` calls. Every movement you write multiplies by it. That single habit is what makes a game run at the same speed on a 30 Hz laptop and a 144 Hz monitor.

```
#include "../Nova.au3"

Global $g_hPerso = 0
Global $g_X = 434, $g_Y = 420, $g_FaceDroite = 1

Func Update($fDT)
    Local $fV = 0
    If _Nova_Key($NOVA_VK_LEFT) Or _Nova_Key(0x41) Then $fV = -1 ; A
    If _Nova_Key($NOVA_VK_RIGHT) Or _Nova_Key(0x44) Then $fV = 1 ; D
    $g_X = _Nova_Clamp($g_X + $fV * 320 * $fDT, 0, 900 - 32)
    If $fV <;> 0 Then $g_FaceDroite = ($fV > 0 ? 1 : 0)
    If _Nova_KeyHit($NOVA_VK_ESCAPE) Then _Nova_Quit()
EndFunc

Func Draw()
    _Nova_ClearColor(_Nova_RGB(28, 42, 62))
    _Nova_FillRect(0, 452, 900, 68, _Nova_RGB(72, 108, 74))
    _Nova_Image($g_hPerso, $g_X, $g_Y, 1.0, Not $g_FaceDroite, 0)
EndFunc

If Not _Nova_Create("My game", 900, 520) Then
    Exit MsgBox(48, "Nova", _Nova_LastError())
```

```
EndIf
$g_hPerso = _Nova_LoadImage(@ScriptDir &; "\assets\player.png")
If $g_hPerso = 0 Then
    MsgBox(48, "Nova", _Nova_LastError())
    _Nova_Destroy()
    Exit
EndIf

_Nova_Run("Update", "Draw", 60)
_Nova_ImageFree($g_hPerso)
_Nova_Destroy()
```

Three input tools, and the difference between them is the whole lesson:

Call	True when	Use it for
`_Nova_Key($vk)`	the key is down right now	walking, holding
`_Nova_KeyHit($vk)`	the key went down **this** step	jumping, menus, one-shot actions
`_Nova_KeyUp($vk)`	the key was released this step	charging, releasing

Letters have no `$NOVA_VK_` constant: they are ASCII codes, so `A` is `0x41` and `D` is `0x44`. `_Nova_Clamp($v, $min, $max)` keeps the character inside the window. And `Not $g_FaceDroite` mirrors the sprite instead of asking you to draw a second, backwards picture file.

Checkpoint if you wrote `$g_X += 4` instead of `$g_X += 320 * $fDT`, the character moves at 4 pixels per frame - which is 240 px/s at 60 fps and 576 px/s at 144 fps. Never write a fixed step again.

06 Step 4 - make something fall

One item is a row in a table. Ten items are ten rows. The engine does not know what a fruit is: it knows rectangles and circles, and your numbers decide.

```
#include "../Nova.au3"

Global $g_hPerso = 0
Global $g_X = 434, $g_Y = 420
Global $aItem[24][3] ; x, y, speed
Global $g_NItem = 0
Global $g_Prochain = 0.3

Func Update($fDT)
    Local $fV = 0
    If _Nova_Key($NOVA_VK_LEFT) Or _Nova_Key(0x41) Then $fV = -1
    If _Nova_Key($NOVA_VK_RIGHT) Or _Nova_Key(0x44) Then $fV = 1
    $g_X = _Nova_Clamp($g_X + $fV * 320 * $fDT, 0, 900 - 32)

    $g_Prochain -= $fDT
    If $g_Prochain <= 0 Then
        If $g_NItem < UBound($aItem) Then
            $aItem[$g_NItem][0] = _Nova_Random(12, 900 - 34)
            $aItem[$g_NItem][1] = -22
            $aItem[$g_NItem][2] = 165
            $g_NItem += 1
        EndIf
        $g_Prochain = 0.75
    EndIf

    Local $i = 0
    While $i < $g_NItem
        $aItem[$i][1] += $aItem[$i][2] * $fDT
        If $aItem[$i][1] > 452 Then ; reached the floor, forget it
            $g_NItem -= 1
            $aItem[$i][0] = $aItem[$g_NItem][0]
            $aItem[$i][1] = $aItem[$g_NItem][1]
            $aItem[$i][2] = $aItem[$g_NItem][2]
        Else
            $i += 1
        EndIf
    WEnd
```

```
    If _Nova_KeyHit($NOVA_VK_ESCAPE) Then _Nova_Quit()
EndFunc

Func Draw()
    _Nova_ClearColor(_Nova_RGB(28, 42, 62))
    _Nova_FillRect(0, 452, 900, 68, _Nova_RGB(72, 108, 74))
    Local $i
    For $i = 0 To $g_NItem - 1
        _Nova_FillCircle($aItem[$i][0] + 11, $aItem[$i][1] + 11, 11, _Nova_RGB(235, 111, 79))
    Next
    _Nova_Image($g_hPerso, $g_X, $g_Y)
EndFunc

If Not _Nova_Create("My game", 900, 520) Then
    Exit MsgBox(48, "Nova", _Nova_LastError())
EndIf
$g_hPerso = _Nova_LoadImage(@ScriptDir &; "\assets\player.png")
If $g_hPerso = 0 Then
    MsgBox(48, "Nova", _Nova_LastError())
    _Nova_Destroy()
    Exit
EndIf
_Nova_Seed(20261105)

_Nova_Run("Update", "Draw", 60)
_Nova_ImageFree($g_hPerso)
_Nova_Destroy()
```

The two habits worth keeping:

- **The table shrinks by swapping with the last row.** When item `$i` dies, the

row that was last moves into its place and the count drops by one. The item that just moved in has not been moved this frame, so you do **not** advance `$i` after a removal - that is exactly why the increment sits in the `Else`.

- `_Nova_Seed` **before** `_Nova_Random`**.** `_Nova_Random($min, $max)` is the

engine's own generator. Seeded, the same fruit falls in the same order every run, which is what makes a game testable. `AutoIt`'s own `Random()` is different on every run and will make your tests lie.

07 Step 5 - catch it, and count it

A catch is two rectangles that overlap. `_Nova_BoxOverlap` is the engine's copy of that sentence, so you do not write four comparisons yourself.

```
#include "../Nova.au3"

Global $g_hPerso = 0
Global $g_X = 434, $g_Y = 420
Global $aItem[24][3] ; x, y, speed
Global $g_NItem = 0
Global $g_Prochain = 0.3
Global $g_Score = 0

Func Update($fDT)
    Local $fV = 0
    If _Nova_Key($NOVA_VK_LEFT) Or _Nova_Key(0x41) Then $fV = -1
    If _Nova_Key($NOVA_VK_RIGHT) Or _Nova_Key(0x44) Then $fV = 1
    $g_X = _Nova_Clamp($g_X + $fV * 320 * $fDT, 0, 900 - 32)

    $g_Prochain -= $fDT
    If $g_Prochain <;= 0 Then
        If $g_NItem <; UBound($aItem) Then
            $aItem[$g_NItem][0] = _Nova_Random(12, 900 - 34)
            $aItem[$g_NItem][1] = -22
            $aItem[$g_NItem][2] = 165 + _Nova_Min(170, $g_Score * 6)
            $g_NItem += 1
        EndIf
        $g_Prochain = 0.75 - _Nova_Min(0.40, $g_Score * 0.012)
    EndIf

    Local $i = 0
    While $i <; $g_NItem
        $aItem[$i][1] += $aItem[$i][2] * $fDT
        If _Nova_BoxOverlap($aItem[$i][0], $aItem[$i][1], 22, 22, $g_X, $g_Y, 32, 32) Then
```

```
            $g_Score += 1
            __Remove($i)
        ElseIf $aItem[$i][1] >; 452 Then
            __Remove($i)
        Else
            $i += 1
        EndIf
    WEnd

    If _Nova_KeyHit($NOVA_VK_ESCAPE) Then _Nova_Quit()
EndFunc

Func __Remove($i)
    Local $j = $g_NItem - 1
    If $i <;> $j Then
        $aItem[$i][0] = $aItem[$j][0]
        $aItem[$i][1] = $aItem[$j][1]
        $aItem[$i][2] = $aItem[$j][2]
    EndIf
    $g_NItem = $j
EndFunc

Func Draw()
    _Nova_ClearColor(_Nova_RGB(28, 42, 62))
    _Nova_FillRect(0, 452, 900, 68, _Nova_RGB(72, 108, 74))
    Local $i
    For $i = 0 To $g_NItem - 1
        _Nova_FillCircle($aItem[$i][0] + 11, $aItem[$i][1] + 11, 11, _Nova_RGB(235, 111, 79))
    Next
    _Nova_Image($g_hPerso, $g_X, $g_Y)
    _Nova_RectUI(8, 8, 210, 34, _Nova_RGBA(0, 0, 0, 140))
    _Nova_TextUI("score " &; $g_Score, 18, 15, _Nova_RGB(255, 255, 255), 17)
EndFunc

If Not _Nova_Create("My game", 900, 520) Then
    Exit MsgBox(48, "Nova", _Nova_LastError())
EndIf
$g_hPerso = _Nova_LoadImage(@ScriptDir &; "\assets\player.png")
If $g_hPerso = 0 Then
    MsgBox(48, "Nova", _Nova_LastError())
    _Nova_Destroy()
    Exit
EndIf
_Nova_Seed(20261105)

_Nova_Run("Update", "Draw", 60)
_Nova_ImageFree($g_hPerso)
_Nova_Destroy()
```

The overlap test takes two boxes: `x, y, w, h` of the fruit and `x, y, w, h` of the character. The fruit is a 22-pixel box even though the circle is drawn from a centre - the drawing and the physics use the same numbers, they just disagree about what "position" means. Read `_Nova_FillCircle($cx, $cy, $r)` as centre, read everything else as top-left corner.

`_Nova_TextUI` and `_Nova_RectUI` are the screen-space twins of `_Nova_Text` and `_Nova_FillRect`: they ignore the camera. A score that scrolls away with the world is a bug, so the HUD always uses the UI pair. `_Nova_RGBA(0, 0, 0, 140)` is black at 140/255 opacity, which is how you get a panel you can still see through.

08 Step 6 - losing, and the two numbers that change the game

Add `$g_Vies = 3`, and make a miss cost one life. Two small changes to `Update`, plus a difficulty curve you already wrote without noticing: the speed `165 + _Nova_Min(170, $g_Score * 6)` and the interval `0.75 - _Nova_Min(0.40, $g_Score * 0.012)` both grow with the score. `_Nova_Min` caps them, so the game gets hard but never impossible.

```
Global $g_Score = 0, $g_Vies = 3
...
        ElseIf $aItem[$i][1] >; 452 Then
            $g_Vies -= 1
            __Remove($i)
            If $g_Vies <;= 0 Then _Nova_Quit()
```

```
        Else
```

and in the HUD:

```
    _Nova_TextUI("score " &; $g_Score & "  lives " & $g_Vies, 18, 15, _Nova_RGB(255, 255, 255), 17)
```

Quitting the whole window when the last life goes is honest for a first game and wrong for a real one. A real game changes **state**: play, then game over, then back to the title. That is Step 7.

09 Step 7 - three screens, one number

A state machine in a small game is one integer and one `Select`. Nothing else.

```
Global $g_Etat = 0 ; 0 title, 1 playing, 2 game over

Func Update($fDT)
    If _Nova_KeyHit($NOVA_VK_ESCAPE) Then _Nova_Quit()

    If $g_Etat <;> 1 Then
        If _Nova_KeyHit($NOVA_VK_ENTER) Or _Nova_MouseHit(1) Then
            If $g_Etat = 2 Then
                $g_Etat = 0 ; game over -> title
            Else
                ResetGame()
                $g_Etat = 1 ; title -> playing
            EndIf
        EndIf
        Return ; no physics on a screen
    EndIf
    ; ... the whole Step 6 body goes on here ...
EndFunc
```

and at the end of `Draw`, painted over the frame:

```
    If $g_Etat <;> 1 Then
        _Nova_RectUI(0, 0, 900, 520, _Nova_RGBA(6, 10, 16, 170))
        If $g_Etat = 0 Then
            _Nova_TextUI("CATCH", 342, 150, _Nova_RGB(255, 214, 90), 46)
            _Nova_TextUI("press ENTER", 398, 262, _Nova_RGB(120, 220, 255), 20)
        Else
            _Nova_TextUI("GAME OVER", 346, 160, _Nova_RGB(235, 111, 79), 40)
            _Nova_TextUI("score " &; $g_Score, 404, 216, _Nova_RGB(255, 255, 255), 24)
        EndIf
    EndIf
```

Three rules make this work, and they are worth stating out loud:

1. `Return` before the physics means a screen cannot lose lives behind the veil.
2. The veil is painted **after** the game frame, so the frame stays visible under the text. That is why the screen feels part of the game and not a dialog.
1. `ResetGame()` is a function, not five lines copy-pasted. Every entry into the game - from the title or from the game over - runs the same reset, so a stale `$g_NItem` can never leak into a new game.

Trap	a key on a menu is usually `_Nova_KeyHit`, never `_Nova_Key`. With `_Nova_Key` the frame after the state change still sees the key down, and your game skips straight from the title into game over.

Nova also has a real scene stack - `_Nova_ScenePush("menu")`, `_Nova_ScenePop()`, and `_Nova_Run("_Nova_SceneUpdate", "_Nova_SceneDraw")` - which is what `02_plateforme.au3` uses when screens must keep their own state. For three screens, one integer is simpler and easier to debug.

10 Step 8 - juice: particles and sound

Two calls that cost a minute each and change how the game feels.

```
; when a fruit is caught
_Nova_ParticleEmit($aItem[$i][0] + 11, $aItem[$i][1] + 11, 16, _
        _Nova_RGB(255, 242, 200), 150, 0.5, 4, 480)
;          centre of the fruit        count  colour       speed  life size gravity

; once per logic step, after the item loop
_Nova_ParticleUpdate($fDT)

; once per frame, after the sprites
_Nova_ParticleDraw()
```

`_Nova_ParticleUpdate` must be in `Update` and `_Nova_ParticleDraw` in `Draw`, for the same reason as everything else: the life of a particle is a number, and numbers live in `Update`.

Sound goes through the Windows multimedia engine, so a `.wav` is the safe choice. Open once, play as often as you like, close at the end:

```
Global $g_SonBien = "", $g_SonMal = ""
$g_SonBien = _Nova_SoundOpen(@ScriptDir &; "\assets\coin.wav")
$g_SonMal = _Nova_SoundOpen(@ScriptDir &; "\assets\mort.wav")
...
_Nova_SoundPlay($g_SonBien)          ; every catch
_Nova_SoundPlay($g_SonMal, 700)      ; every miss, at 70 % volume
...
_Nova_SoundClose($g_SonBien)
_Nova_SoundClose($g_SonMal)
```

`_Nova_SoundOpen` returns an alias **string**, and an empty string on failure - `_Nova_SoundPlay("")` is then a harmless no-op, so a machine without a sound card does not crash your game.
`_Nova_MusicPlay("theme.mp3")` plays background music on the same machinery.

11 Step 9 - a game that tests itself

The last step is not a feature: it is proof that the eight before it work. A `/check` mode makes the game play itself, capture one frame, and assert its pixels. It is the reason you can change code at 2 a.m. and still know the game runs.

```
Global $g_Check = 0, $g_Limite = 310, $g_Pas = 0
$g_Check = ($CmdLine[0] >; 0 And $CmdLine[1] = "/check") ? 1 : 0

Func Update($fDT)
    $g_Pas += 1
    If $g_Check Then $fV = __Steer() ; follow the lowest item, like a player would
    ...
    If $g_Check And $g_Pas >;= $g_Limite Then
        _Nova_Capture(@ScriptDir &; "\_catch.png")
        _Nova_Quit()
    EndIf
EndFunc

Func __Steer()
    Local $i, $fBas = -1, $iBas = -1
    For $i = 0 To $g_NItem - 1
        If $aItem[$i][1] >; $fBas Then
            $fBas = $aItem[$i][1]
            $iBas = $i
        EndIf
    Next
    If $iBas <; 0 Then Return 0
    Local $fCentre = $g_X + 16, $fCible = $aItem[$iBas][0] + 11
    If Abs($fCible - $fCentre) <; 8 Then Return 0
    Return ($fCible >; $fCentre) ? 1 : -1
EndFunc
```

and after the loop, a one-line report and an exit code:

```
Local $h = FileOpen(@ScriptDir &; "\_catch_out.txt", 2 + 8)
FileWriteLine($h, "pas=" &; $g_Pas & " score=" & $g_Score & " vies=" & $g_Vies)
FileClose($h)
Exit ($g_Pas >;= $g_Limite And $g_Score > 0) ? 0 : 1
```

Four decisions in there, and each one exists because the alternative was a lie:

Decision	Why
Capture in `Update`	`_Nova_Capture` replays the scene on its own target; from `Draw` it is inside a frame that is already open.
Trigger on a step count, not on `_Nova_Time()`	wall time depends on the machine. Steps do not. Measured: the 32-bit and 64-bit runs of `03_catch.au3 /check` agree to 0 pixels out of 468 000.
Steer the character by script	a score of 0 would leave the catch branch unproven, and a green suite would be hiding a broken game.
Exit code `0` only when the score moved	the number *is* the test. Anything else is a screenshot someone hopes at.

Then assert the picture, in `tests/03_catch.txt`, one line per claim:

```
size 900x520
at 8,68 #1c2a3e
at 300,451 #78a06e
count #486c4a 0,455,899,519 58439 8 2
count #e84c38 0,418,899,451 144 8 2
none #ffd65a
```

Read it as sentences: the gap between two sky bands is the background colour, the green line is where the floor starts, exactly 58 439 floor pixels survive in the bottom band (the rest is eaten by a particle spray), the red shirt of the character is 144 pixels, and no gold star is on screen at step 310. Every number is computed from the program, not from how the picture "looks".

Run the whole engine's proof with one command from the package root:

```
python tools/nova_suite.py
```

Its exit code is the number of failed checks, so `0` means everything passed: syntax, source hygiene, COM-call arity, the writing template, this course's PDF, six programs, three image locks - 14 steps, on the interpreter you point it at.

12 The finished program

This is `examples/03_catch.au3` exactly as it ships, all 11 functions. It is the course's nine steps assembled: the character and its animation from Step 2, the walking from Step 3, the falling table from Step 4, the catch from Step 5, the lives from Step 6, the three screens from Step 7, the juice from Step 8, and the self-test from Step 9.

examples/03_catch.au3

```
; #INDEX# =======================================================================================================================
; Title .........: Nova - catch game example
; AutoIt Version : 3.3.16.1
; Language ......: English
; Description ...: The game that docs/course.md builds step by step: walk under
;                  falling fruit, catch it, keep your three lives, and read the
;                  score. Run it with /check to drive it by script, capture a PNG
;                  and assert the loop.
; Author(s) .....: Numeric1
; Copyright .....: (C) 2026 Numeric1
; License .......: MIT - see the LICENSE file at the package root
; ===============================================================================================================================
#include "../Nova.au3"


; #CURRENT# =====================================================================================================================
;Draw
;ResetGame
;SpawnItem
;Update
; ===============================================================================================================================

; #INTERNAL# ====================================================================================================================
;__Catch_Ecran
;__Catch_Remove
;__Catch_Steer
; ===============================================================================================================================

; --- what the game looks like ------------------------------------------------
Global Const $g_iSol = 452  ; the top edge of the floor, in pixels
Global Const $g_iTaille = 32  ; the character is one 32x32 frame of player.png
```

```
; --- what the game remembers ---------------------------------------------------
Global $g_hPerso = 0 ; sprite handle, 0 = not loaded
Global $g_SonBien = "" ; alias of the catch sound
Global $g_SonMal = "" ; alias of the miss sound
Global $g_X = 434, $g_Y = 400 ; character position, top-left corner
Global $g_FaceDroite = 1 ; which way the character looks
```

examples/03_catch.au3 (line 39 of 365)

```
Global $g_Score = 0, $g_Vies = 3
Global $g_Etat = 0 ; 0 title screen, 1 playing, 2 game over
Global $g_Prochain = 0 ; seconds before the next item appears
Global $aItem[24][4] ; per item: x, y, speed, kind (0 fruit, 1 star)
Global $g_NItem = 0 ; how many items are on screen

; --- the scripted gate ---------------------------------------------------------
Global $g_Check = 0, $g_Limite = 0, $g_CaptureFaite = -1 ; -1 = capture not tried yet
Global $g_Pas = 0 ; logic steps taken, the clock the scripted gate reads
Global $g_Temps = 0 ; the game clock, in seconds, built from $fDT only

; #FUNCTION# ====================================================================================================================
; Name...........: ResetGame
; Description ...: Puts every number back to the value a new game starts with.
; Syntax.........: ResetGame()
; Parameters ....:
; Return values .:
; Author ........: Numeric1
; Modified.......:
; Remarks .......: The seed is part of the reset: with the same seed the same fruit
;                  falls in the same order, so a bug can be shown twice.
; Related .......: _Nova_Seed
; Link ..........:
; Example .......:
; ===============================================================================================================================
Func ResetGame()
■$g_Score = 0
■$g_Vies = 3
■$g_X = (_Nova_Width() - $g_iTaille) / 2
■$g_Y = $g_iSol - $g_iTaille
■$g_NItem = 0
■$g_Prochain = 0.3
■_Nova_ParticleClear()
■_Nova_Seed(20261105)
■$g_Pas = 0
■$g_Temps = 0
EndFunc   ;==>ResetGame
```

examples/03_catch.au3 (line 77 of 365)

```
; #FUNCTION# ====================================================================================================================
; Name...........: SpawnItem
; Description ...: Adds one item at the top of the screen, at a random x.
; Syntax.........: SpawnItem()
; Parameters ....:
; Return values .: Success: index of the new item. Failure: -1, the pool is full.
; Author ........: Numeric1
; Modified.......:
; Remarks .......: One star in six is worth five points and falls faster; it is the
;                  only reason to look at two kinds of item.
; Related .......: _Nova_Random
; Link ..........:
; Example .......:
; ================================================================================================================================
Func SpawnItem()
■If $g_NItem >;= UBound($aItem) Then Return -1
■Local $i = $g_NItem
■$aItem[$i][3] = (_Nova_Random() <; 0.17) ? 1 : 0 ; kind first, the speed uses it
■$aItem[$i][0] = _Nova_Random(12, _Nova_Width() - 34)
■$aItem[$i][1] = -22
■$aItem[$i][2] = ($aItem[$i][3] = 1 ? 300 : 165 + _Nova_Min(170, $g_Score * 6))
■$g_NItem += 1
■Return $i
EndFunc   ;==>SpawnItem

; #FUNCTION# ====================================================================================================================
; Name...........: Update
; Description ...: One logic step: read the keys, move, drop items, score, lives.
; Syntax.........: Update($fDT)
; Parameters ....: $fDT - seconds elapsed since the last logic step
; Return values .:
; Author ........: Numeric1
; Modified.......:
; Remarks .......: $g_Etat decides which rules run, so the title screen and the game
;                  over screen cost nothing while the game is being played.
; Related .......: Draw
; Link ..........:
; Example .......:
```

examples/03_catch.au3 (line 115 of 365)

```
; ===============================================================================================================================
Func Update($fDT)
■$g_Pas += 1
■$g_Temps += $fDT
■If _Nova_KeyHit($NOVA_VK_ESCAPE) Then _Nova_Quit()

■; --- the two screens that are not the game itself ---------------------------
■If $g_Etat <;> 1 Then
■■If _Nova_KeyHit($NOVA_VK_ENTER) Or _Nova_KeyHit($NOVA_VK_SPACE) Or _Nova_MouseHit(1) Then
■■■ResetGame()
■■■$g_Etat = ($g_Etat = 2 ? 0 : 1) ; game over goes back to the title
■■EndIf
■Else
■■; --- walking ----------------------------------------------------------------
■■Local $fV = 0
■■If _Nova_Key($NOVA_VK_LEFT) Or _Nova_Key(0x41) Then $fV = -1 ; A
■■If _Nova_Key($NOVA_VK_RIGHT) Or _Nova_Key(0x44) Then $fV = 1 ; D
■■If $g_Check Then $fV = __Catch_Steer() ; the scripted driver follows the lowest item
■■$g_X = _Nova_Clamp($g_X + $fV * 320 * $fDT, 0, _Nova_Width() - $g_iTaille)
■■If $fV <;> 0 Then $g_FaceDroite = ($fV > 0 ? 1 : 0)

■■; --- one item every 0.75 s, a little faster as the score grows ----------
■■$g_Prochain -= $fDT
■■If $g_Prochain <;= 0 Then
■■■SpawnItem()
■■■$g_Prochain = 0.75 - _Nova_Min(0.40, $g_Score * 0.012)
■■EndIf

■■; --- falling, catching, missing ---------------------------------------
■■; an item that leaves the pool shifts the ones after it down, so the index
■■; is only advanced when nothing was removed
■■Local $i = 0
■■While $i <; $g_NItem
■■■$aItem[$i][1] += $aItem[$i][2] * $fDT
■■■If _Nova_BoxOverlap($aItem[$i][0], $aItem[$i][1], 22, 22, $g_X, $g_Y, $g_iTaille, $g_iTaille) Then
■■■■$g_Score += ($aItem[$i][3] = 1 ? 5 : 1)
■■■■_Nova_ParticleEmit($aItem[$i][0] + 11, $aItem[$i][1] + 11, 16, _
■■■■■■_Nova_RGB(255, 242, 200), 150, 0.5, 4, 480)
```

examples/03_catch.au3 (line 153 of 365)

```
■■■■_Nova_SoundPlay($g_SonBien)
■■■■__Catch_Remove($i)
■■■ElseIf $aItem[$i][1] >; $g_iSol Then
■■■■$g_Vies -= 1
■■■■_Nova_ParticleEmit($aItem[$i][0] + 11, $g_iSol - 4, 10, _Nova_RGB(210, 70, 70), 120, 0.4, 3, 420)
■■■■_Nova_SoundPlay($g_SonMal)
■■■■__Catch_Remove($i)
■■■■If $g_Vies <;= 0 Then $g_Etat = 2
■■■Else
■■■■$i += 1
■■■EndIf
■■WEnd

■■_Nova_ParticleUpdate($fDT)
■EndIf

■; in check mode the capture is taken INSIDE the logic : _Nova_Capture replays
■; the scene on its own target, a call inside Draw() would be wrong. The trigger
■; is a step count, not a clock, so the same frame is captured on a 30 Hz laptop,
■; on 144 Hz, on the 32-bit and on the 64-bit interpreter
■If $g_Check And $g_CaptureFaite <; 0 And $g_Pas >= $g_Limite Then
■■$g_CaptureFaite = _Nova_Capture(@ScriptDir &; "\_catch.png")
■■_Nova_Quit()
■EndIf
EndFunc   ;==>Update

; #FUNCTION# ====================================================================================================================
; Name...........: Draw
; Description ...: Paints one frame: sky, floor, items, character, interface.
; Syntax.........: Draw()
; Parameters ....:
; Return values .:
; Author ........: Numeric1
; Modified.......:
; Remarks .......: Frame 3 of the 4-frame sheet comes from _Nova_AnimFrame, so the
;                  character walks without a single line of animation bookkeeping.
; Related .......: Update
; Link ..........:
```

examples/03_catch.au3 (line 191 of 365)

```
; Example .....:
; ==============================================================================================================================
Func Draw()
■_Nova_ClearColor(_Nova_RGB(28, 42, 62))

■; sky bands and floor, plain rectangles
■Local $i
■For $i = 0 To 5
■■_Nova_FillRect(0, $i * 72, _Nova_Width(), 66, _Nova_RGB(30 + $i * 4, 46 + $i * 5, 68 + $i * 6))
■Next
■_Nova_FillRect(0, $g_iSol, _Nova_Width(), _Nova_Height() - $g_iSol, _Nova_RGB(72, 108, 74))
■_Nova_DrawLine(0, $g_iSol, _Nova_Width(), $g_iSol, _Nova_RGB(120, 160, 110), 2.0)

■; items : a fruit is an orange ball, a star is the same ball in gold and smaller
■For $i = 0 To $g_NItem - 1
■■If $aItem[$i][3] = 1 Then
■■■_Nova_FillCircle($aItem[$i][0] + 11, $aItem[$i][1] + 11, 9, _Nova_RGB(255, 214, 90))
■■■_Nova_DrawEllipse($aItem[$i][0] + 11, $aItem[$i][1] + 11, 13, 13, _Nova_RGB(255, 244, 180), 1.5)
■■Else
■■■_Nova_FillCircle($aItem[$i][0] + 11, $aItem[$i][1] + 11, 11, _Nova_RGB(235, 111, 79))
■■EndIf
■Next

■; the character : one frame of a 128x32 sheet, mirrored when walking left
■_Nova_ImagePart($g_hPerso, _Nova_AnimFrame($g_Temps, 4, 8) * 32, 0, 32, 32, _
■■■$g_X, $g_Y, -1, -1, 1.0, Not $g_FaceDroite, 0)
■_Nova_ParticleDraw()

■; interface : RectUI/TextUI ignore the camera, so they never drift
■_Nova_RectUI(8, 8, 210, 34, _Nova_RGBA(0, 0, 0, 140))
■_Nova_TextUI("score " &; $g_Score & "  lives " & $g_Vies, 18, 15, _Nova_RGB(255, 255, 255), 17)
■; in check mode the corner reads the step count, not the fps: a live number
■; makes the captured png differ from run to run, and a png that is not the same
■; bytes twice cannot print the same PDF twice, so the delivery zip could never
■; have one stable fingerprint. The fps still reaches the bilan, in text.
■_Nova_TextUI("Nova " &; _Nova_Version() & "  -  " & ($g_Check ? ($g_Pas & " pas") : (Int(_Nova_FPS()) & " IPS")), _
■■■_Nova_Width() - 190, 15, _Nova_RGB(170, 190, 210), 13)
```

examples/03_catch.au3 (line 229 of 365)

```
■If $g_Etat <;> 1 Then __Catch_Ecran()
EndFunc   ;==>Draw

; #FUNCTION# ====================================================================================================================
; Name...........: __Catch_Remove
; Description ...: Drops item $i from the pool by moving the last one into its place.
; Syntax.........: __Catch_Remove($i)
; Parameters ....: $i - index of the item to remove
; Return values .:
; Author ........: Numeric1
; Modified.......:
; Remarks .......: Order does not matter here, so the cheap swap beats shifting every
;                  row down by one.
; Related .......: Update
; Link ..........:
; Example .......:
; ================================================================================================================================
Func __Catch_Remove($i)
■Local $j = $g_NItem - 1
■If $i <;> $j Then
■■$aItem[$i][0] = $aItem[$j][0]
■■$aItem[$i][1] = $aItem[$j][1]
■■$aItem[$i][2] = $aItem[$j][2]
■■$aItem[$i][3] = $aItem[$j][3]
■EndIf
■$g_NItem = $j
EndFunc   ;==>__Catch_Remove

; #FUNCTION# ====================================================================================================================
; Name...........: __Catch_Steer
; Description ...: Direction the scripted gate gives to the character.
; Syntax.........: __Catch_Steer()
; Parameters ....:
; Return values .: -1, 0 or 1, the same three values the keyboard can produce.
; Author ........: Numeric1
; Modified.......:
; Remarks .......: /check must catch something to prove the scoring path; a human
;                  does it by playing, the suite does it here.
```

examples/03_catch.au3 (line 267 of 365)

```
; Related .......: Update
; Link ..........:
; Example .......:
; ===============================================================================================================================
Func __Catch_Steer()
■Local $i, $fBas = -1, $iBas = -1
■For $i = 0 To $g_NItem - 1
■■If $aItem[$i][1] >; $fBas Then
■■■$fBas = $aItem[$i][1]
■■■$iBas = $i
■■EndIf
■Next
■If $iBas <; 0 Then Return 0
■Local $fCentre = $g_X + $g_iTaille / 2
■Local $fCible = $aItem[$iBas][0] + 11
■If Abs($fCible - $fCentre) <; 8 Then Return 0
■Return ($fCible >; $fCentre) ? 1 : -1
EndFunc   ;==>__Catch_Steer

; #FUNCTION# ====================================================================================================================
; Name...........: __Catch_Ecran
; Description ...: Paints the title screen or the game over screen over the frame.
; Syntax.........: __Catch_Ecran()
; Parameters ....:
; Return values .:
; Author ........: Numeric1
; Modified.......:
; Remarks .......: A veil rectangle first, then the text : the frame behind stays
;                  readable, which is what makes the screen feel part of the game.
; Related .......: Draw
; Link ..........:
; Example .......:
; ===============================================================================================================================
Func __Catch_Ecran()
■Local $iMilieu = _Nova_Width() / 2
■_Nova_RectUI(0, 0, _Nova_Width(), _Nova_Height(), _Nova_RGBA(6, 10, 16, 170))
■If $g_Etat = 0 Then
■■_Nova_TextUI("CATCH", $iMilieu - 78, 150, _Nova_RGB(255, 214, 90), 46)
```

examples/03_catch.au3 (line 305 of 365)

```
■■_Nova_TextUI("arrows or A / D to walk, catch the fruit, keep three lives", _
■■■■$iMilieu - 258, 214, _Nova_RGB(220, 230, 240), 17)
■■_Nova_TextUI("press ENTER", $iMilieu - 52, 262, _Nova_RGB(120, 220, 255), 20)
■Else
■■_Nova_TextUI("GAME OVER", $iMilieu - 104, 160, _Nova_RGB(235, 111, 79), 40)
■■_Nova_TextUI("score " &; $g_Score, $iMilieu - 46, 216, _Nova_RGB(255, 255, 255), 24)
■■_Nova_TextUI("press ENTER to go back to the title", $iMilieu - 152, 258, _Nova_RGB(200, 215, 230), 16)
■EndIf
EndFunc   ;==>__Catch_Ecran

$g_Check = ($CmdLine[0] >; 0 And ($CmdLine[1] = "/check" Or $CmdLine[1] = "//check")) ? 1 : 0
If $g_Check Then _Nova_SetLog(@ScriptDir &; "\_catch_log.txt", 2)

If Not _Nova_Create("Nova - catch", 900, 520, 0) Then
■MsgBox(16, "Nova", "cannot start the engine:" &; @LF & _Nova_LastError())
■Exit 2
EndIf

; assets live beside the example, so the zip can be unzipped anywhere
$g_hPerso = _Nova_LoadImage(@ScriptDir &; "\assets\player.png")
If $g_hPerso = 0 Then
■MsgBox(16, "Nova", "cannot load assets\player.png:" &; @LF & _Nova_LastError())
■_Nova_Destroy()
■Exit 3
EndIf
$g_SonBien = _Nova_SoundOpen(@ScriptDir &; "\assets\coin.wav")
$g_SonMal = _Nova_SoundOpen(@ScriptDir &; "\assets\mort.wav")

ResetGame()
$g_Limite = ($g_Check ? 310 : 0) ; step the capture is taken at, see Update()
If $g_Check Then
■; the scripted driver starts the game the way the title screen tells a human
■; to, and one Update() call consumes the key press outside the measured frames
■_Nova_InputScript(1)
■_Nova_KeyInject($NOVA_VK_ENTER, 1)
■Update(1.0 / 60)
■_Nova_KeyInject($NOVA_VK_ENTER, 0)
EndIf
```

examples/03_catch.au3 (line 343 of 365)

```
_Nova_Run("Update", "Draw", 60)

Local $iFrames = _Nova_FrameCount()
If $g_Check Then
■; bilan = one line of key=value for tools/nova_suite.py
■Local $fFps = Round(_Nova_FPS(), 1)
■Local $h = FileOpen(@ScriptDir &; "\_catch_out.txt", 2 + 8)
■If $h <;> -1 Then
■■FileWriteLine($h, "images=" &; $iFrames & " pas=" & $g_Pas & " ips=" & $fFps & " score=" & $g_Score & _
■■■■" vies=" &; $g_Vies & " etat=" & $g_Etat & " capture=" & $g_CaptureFaite)
■■FileClose($h)
■EndIf
■_Nova_Destroy()
■; the scripted driver has to score, otherwise the catch branch is not proven;
■; the fps window is the one measured on a 60 Hz panel, see 01_hello.au3
■Exit ($g_Pas >;= $g_Limite And $g_CaptureFaite And $g_Score > 0 And $g_Vies = 3) ? 0 : 1
EndIf

_Nova_ImageFree($g_hPerso)
_Nova_SoundClose($g_SonBien)
_Nova_SoundClose($g_SonMal)
_Nova_Destroy()
```

Where to look once you are in it:

You want	Go to
The loop and its two callbacks	`Update`, `Draw`
The reset every new game runs	`ResetGame`
The only place a fruit is born	`SpawnItem`
The state machine and the screens	top of `Update`, `__Catch_Ecran`
The physics and the catch test	the `While $i < $g_NItem` block
The self-test	the bottom of the file, `$g_Check`

13 Make it yours

Each line is one feature, one or two functions, and a hint. Do them in order; they are graded like exercises, not like a wish list.

#	Change	Hint
1	Show the best score after the game over	`_Nova_SaveSet("best", ...)` and `_Nova_SaveWrite`, see the save functions
2	A second kind of fruit worth 5 points	you already have `$aItem[][3]` as a kind column - compare it in `Draw` and in the score
3	Freeze the character for half a second on a miss	a counter in `Update` that blocks the movement while it is above zero
4	Make the fruit move sideways too	one more column in the table, `_Nova_Random(-60, 60) * $fDT`
5	A paddle instead of a character: mouse control	`_Nova_MouseX()` is already the x you want, and `_Nova_MouseHit(1)` starts the game
6	Play the animation only while walking	`_Nova_AnimFrame($g_Pas / 60.0, 4, 8)` when moving, a fixed frame otherwise
7	A pause screen	a fourth value of `$g_Etat`, `Return` before physics like Step 7
8	Ten rows of level art behind the game	`_Nova_TileCreate` and `_Nova_TileDraw`, see `02_plateforme.au3`
9	A camera that follows the player up a tall level	`_Nova_CameraFollow($x, $y, ...)` - and remember the HUD must use the `_Nova_*UI` calls
10	Your own game with the same skeleton	keep `Update`/`Draw`/`ResetGame`/`$g_Etat` and change what the table means

Try run `examples/03_catch.au3` first, then `02_plateforme.au3`, then `01_hello.au3`. The platformer is the same loop with tiles, a body, a camera and three scenes: it is the map of where you are going.

14 When it does not work

What you see	What it means	What to do
Nothing happens at all	an error dialog you did not see	run `AutoIt3.exe /ErrorStdOut mon_jeu.au3` and read the terminal

What you see	What it means	What to do
`_Nova_Create` **returns 0**	the graphics chain could not start	print `_Nova_LastError()`; try `_Nova_Create("t", 320, 240, 1)` which forces the safe software path
A draw call returns 0 with `@error = 7`	painted outside the frame	move it into `Draw()`
`_Nova_LoadImage` **returns 0**	wrong path or not an image	`@ScriptDir & "\assets\..."` only works when your file is in `examples`
Keys do nothing	window not focused, or you used `_Nova_Key` where you meant `_Nova_KeyHit`	click the window; check the function name
The character is too fast on one PC, slow on another	a movement without `$fDT`	multiply every speed by `$fDT`
The score climbs by itself	`Draw()` is doing arithmetic	numbers change in `Update` only
A black window	nothing painted yet	check the two names you passed to `_Nova_Run`
Your test is green but the game is broken	the test never reached that state	every screen a human can reach needs a scripted trigger, see Step 9

Two lines that solve most mysteries:

```
_Nova_SetLog(@ScriptDir &; "\_log.txt", 2) ; before _Nova_Create, level 2 keeps everything
MsgBox(48, "Nova", _Nova_LastError())       ; after any call that returned 0
```

15 Words this course uses

Word	Meaning here
Frame	one pass of `Draw`, one picture on screen
Step (or tick)	one pass of `Update`, a fixed 1/60 s of game time
`$fDT`	seconds between two steps, handed to you by the loop
Handle	a number that names an engine object, `0` = failure
Sprite sheet	one picture file holding several frames, addressed by source rectangle
Screen space	pixels from the top-left of the window, camera ignored (`_Nova_*UI`)
World space	pixels in the level, camera applied (`_Nova_Text`, `_Nova_FillRect`)
State machine	one integer that decides which rules run this step
Seeded random	a generator whose sequence repeats when the seed repeats
Image lock	a file of pixel assertions that makes a rendering change visible

16 Where to read next

- `docs/quick_start.md` - the same engine in nine sections, one page per idea, including tiles, bodies, jumps and the measured jump height.
- `docs/api.md` - all 122 public functions, generated from the source.
- `docs/technique.md` (French) - the measurements behind the rules: why colours go by address, why a structure passed by value is an ABI hole, what the capture pipeline does.
- `examples/02_plateforme.au3` - a full platformer with a level in text, a

physics body, a camera and three scenes, plus `/route` which proves the level is completable before a human plays it.

- `MANIFEST.md` - which files are required, which are documentation, which are

build products.

Nova 1.0 - MIT - Numeric1. The engine is four `.au3` files and one door; this course is what you do with them.