Real-time Game Physics

Why Physics?

- The Human Experience
  - Real-world motions are physically-based
  - Physics can make simulated game worlds appear more natural
  - Makes sense to strive for physically-realistic motion for some types of games
- Emergent Behavior
  - Physics simulation can enable a richer gaming experience

Why Physics?

- Developer/Publisher Cost Savings
  - Classic approaches to creating realistic motion:
    - Artist-created keyframe animations
    - Motion capture
    - Both are labor intensive and expensive
  - Physics simulation:
    - Motion generated by algorithm
    - Theoretically requires only minimal artist input
    - Potential to substantially reduce content development cost

High-level Decisions

- Physics in Digital Content Creation Software:
  - Many DCC modeling tools provide physics
  - Export physics-engine-generated animation as keyframe data
  - Enables incorporation of physics into game engines that do not support real-time physics
  - Straightforward update of existing asset creation pipelines
  - Does not provide player with the same emergent-behavior-rich game experience
  - Does not provide full cost savings to developer/publisher

High-level Decisions

- Real-time Physics in Game at Runtime:
  - Enables the emergent behavior that provides player a richer game experience
  - Potential to provide full cost savings to developer/publisher
  - May require significant upgrade of game engine
  - May require significant update of asset creation pipelines
  - May require special training for modelers, animators, and level designers
  - Licensing an existing engine may significantly increase third party middleware costs

High-level Decisions

- License vs. Build Physics Engine:
  - License middleware physics engine
    - Complete solution from day 1
    - Proven, robust code base (in theory)
    - Most offer some integration with DCC tools
    - Features are always a tradeoff

High-level Decisions

- License vs. Build Physics Engine:
  - Build physics engine in-house
    - Choose only the features you need
    - Opportunity for more game-specific optimizations
    - Greater opportunity to innovate
    - Cost can be easily be much greater
    - No asset pipeline at start of development

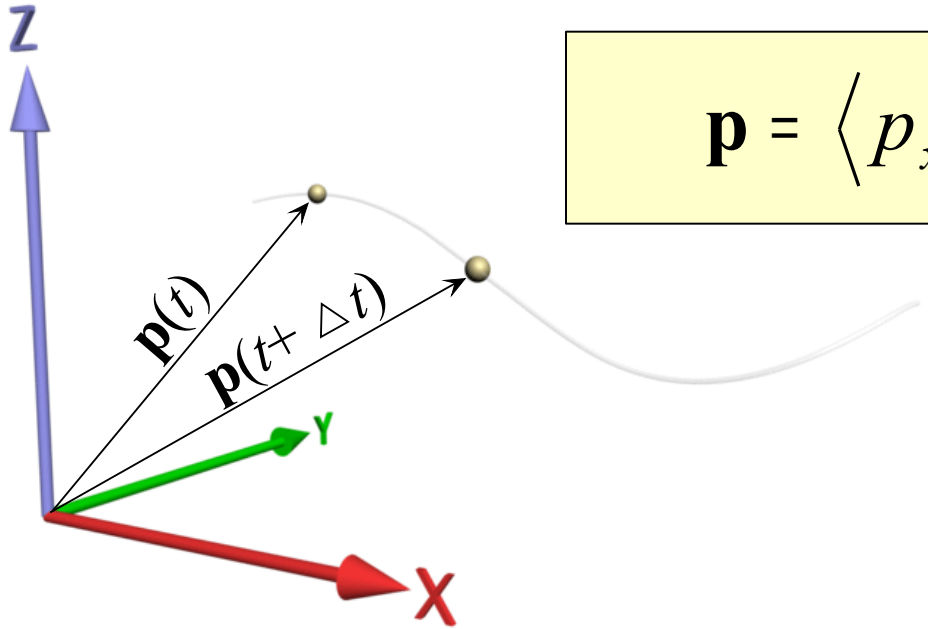
Particle Physics

Particle Physics

- What is a Particle?
  - A sphere of finite radius with a perfectly smooth, frictionless surface
  - Experiences no rotational motion
- Particle Kinematics
  - Defines the basic properties of particle motion
  - Position, Velocity, Acceleration

Particle Kinematics - Position

- Location of Particle in World Space
  - SI Units: meters (m)



$$\mathbf{p} = \langle p_x, p_y, p_z \rangle$$

  - Changes over time when object moves

Particle Kinematics - Velocity and Acceleration

- Velocity (SI units: m/s)
  - First time derivative of position:

$$\mathbf{V}(t) = \lim_{\Delta t \to 0} \frac{\mathbf{p}(t + \Delta t) - \mathbf{p}(t)}{\Delta t} = \frac{d}{dt}\mathbf{p}(t)$$

- Acceleration (SI units: m/s^2)
  - First time derivative of velocity
  - Second time derivative of position

$$\mathbf{a}(t) = \frac{d}{dt}\mathbf{V}(t) = \frac{d^2}{dt^2}\mathbf{p}(t)$$

Newton's 2nd Law of Motion

- Paraphrased – "An object's change in velocity is proportional to an applied force"
- The Classic Equation:

$$\mathbf{F}(t) = m\mathbf{a}(t)$$

  - m = mass (SI units: kilograms, kg)
  - $\mathbf{F}(t)$ = force (SI units: Newtons)

What is Physics Simulation?

- The Cycle of Motion:
  - Force, $\mathbf{F}(t)$, causes acceleration
  - Acceleration, $\mathbf{a}(t)$, causes a change in velocity
  - Velocity, $\mathbf{V}(t)$ causes a change in position
- Physics Simulation:
  - Solving variations of the above equations over time to emulate the cycle of motion

Example: 3D Projectile Motion

- Constant Force
  - Weight of the projectile, $\mathbf{W} = m\mathbf{g}$
  - $\mathbf{g}$ is constant acceleration due to gravity
- Closed-form Projectile Equations of Motion:

$$\mathbf{V}(t) = \mathbf{V}_{init} + \mathbf{g}\left(t - t_{init}\right)$$

$$\mathbf{p}(t) = \mathbf{p}_{init} + \mathbf{V}_{init}\left(t - t_{init}\right) + \frac{1}{2}\mathbf{g}\left(t - t_{init}\right)^2$$

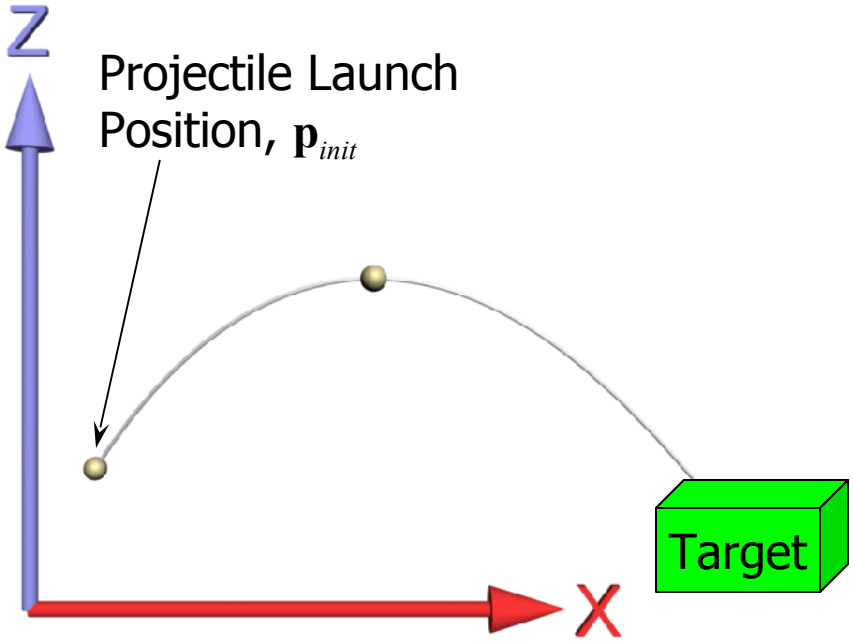
  - These closed-form equations are valid, *and exact**, for any time, t, in seconds, greater than or equal to t_{init}

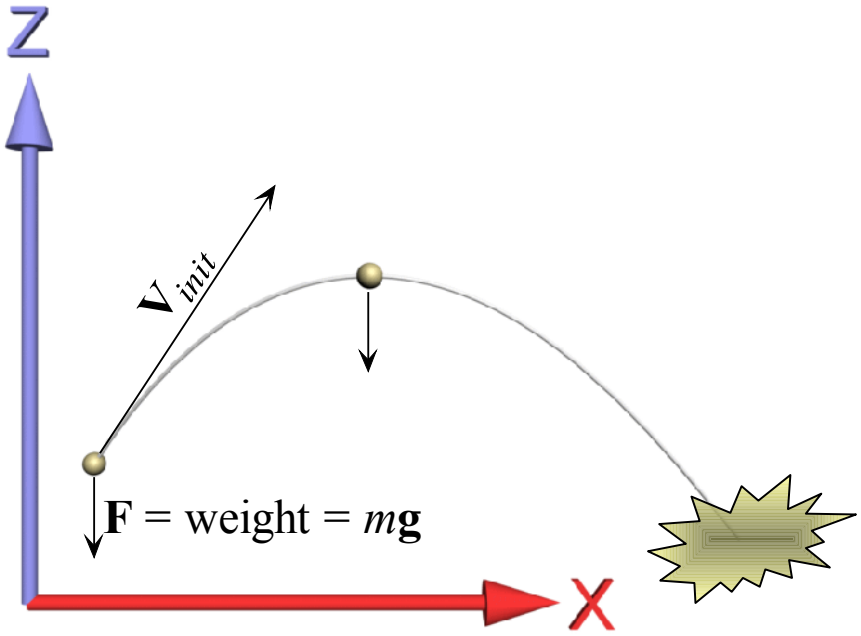
Example: 3D Projectile Motion

- Initial Value Problem
  - Simulation begins at time t_{init}
  - The initial velocity, $\mathbf{V}_{init}$ and position, $\mathbf{p}_{init}$, at time t_{init}, are known
  - Solve for later values at any future time, t, based on these initial values
- On Earth:
  - If we choose positive Z to be straight up (away from center of Earth), g_{Earth} = 9.81 m/s^2:

$$\mathbf{g}_{Earth} = -g_{Earth}\hat{k} = \langle 0.0, 0.0, -9.81 \rangle \text{ m/s}^2$$

Concrete Example: Target Practice





Concrete Example: Target Practice

- Choose $\mathbf{V}_{init}$ to Hit a Stationary Target
  - $\mathbf{p}_{target}$ is the stationary target location
  - We would like to choose the initial velocity, $\mathbf{V}_{init}$, required to hit the target at some future time, t_{hit}.
  - Here is our equation of motion at time t_{hit}:

$$\mathbf{p}_{target} = \mathbf{p}_{init} + \mathbf{V}_{init}\left(t_{hit} - t_{init}\right) + \frac{1}{2}\mathbf{g}\left(t_{hit} - t_{init}\right)^2$$

  - Solution in general is a bit tedious to derive…
  - Infinite number of solutions!
  - Hint: Specify the magnitude of $\mathbf{V}_{init}$, solve for its direction

Concrete Example: Target Practice

- Choose Scalar launch speed, V_{init}, and Let:

$$\mathbf{V}_{init} = \langle V_{init}\cos\theta\ \cos\phi\ , V_{init}\sin\theta\ \cos\phi\ , V_{init}\sin\phi\ \rangle$$

- Where:

$$\cos\theta = \frac{p_{target,x} - p_{init,x}}{\sqrt{\left(p_{target,x} - p_{init,x}\right)^2 - \left(p_{target,y} - p_{init,y}\right)^2}}\ ;\ \sin\theta = \frac{p_{target,y} - p_{init,y}}{\sqrt{\left(p_{target,x} - p_{init,x}\right)^2 - \left(p_{target,y} - p_{init,y}\right)^2}}$$

$$\tan\phi = \frac{A \pm \sqrt{A^2 - 2g\left(\frac{A}{V_{init}}\right)^2\left(\frac{1}{2}g\left(\frac{A}{V_{init}}\right)^2 + p_{target,z} - p_{init,z}\right)}}{g}\left(\frac{V_{init}}{A}\right)^2$$

$$A = \frac{\left(p_{target,y} + p_{target,x}\right) - \left(p_{init,y} + p_{init,x}\right)}{\left(\cos\theta + \sin\theta\right)}$$

Concrete Example: Target Practice

- If Radicand in $\tan\phi$ Equation is Negative:
  - No solution. V_{init} is too small to hit the target

$$\text{if } \left(A^2 - 2g\left(\frac{A}{V_{init}}\right)^2 \left(\frac{1}{2} g \left(\frac{A}{V_{init}}\right)^2 + p_{target,z} - p_{init,z} \right) \right) < 0, \text{ then no solution!}$$

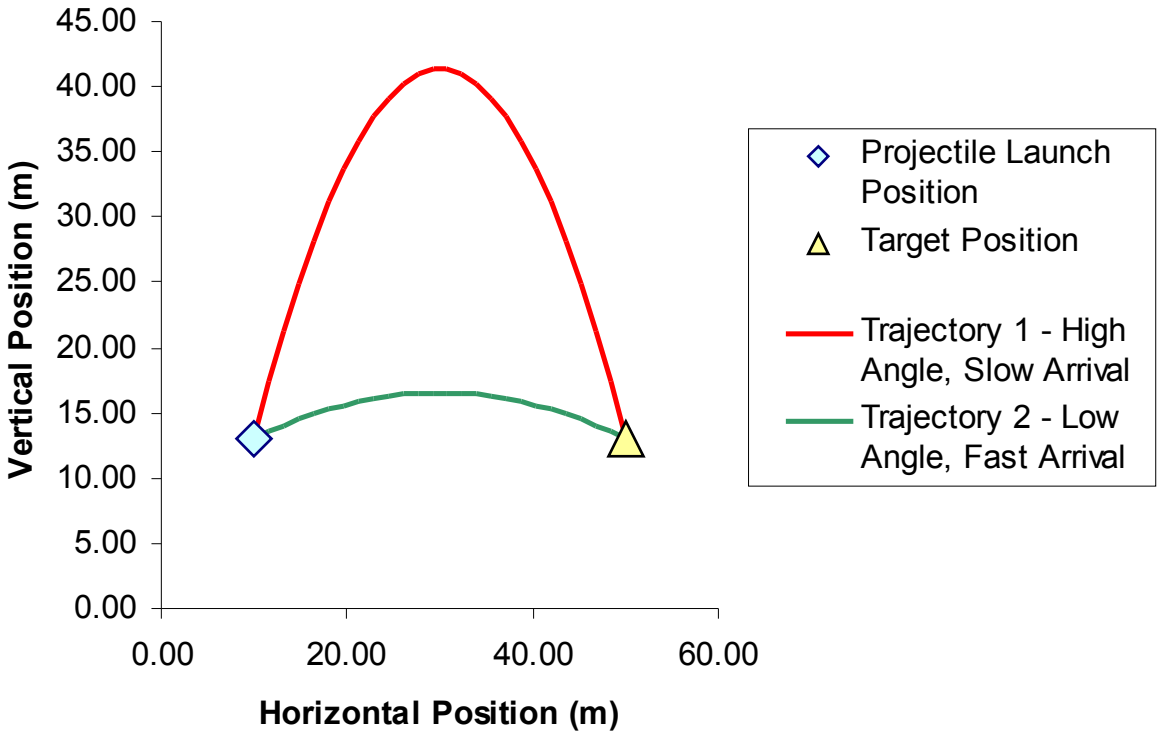
- Otherwise:
  - One solution if radicand == 0
  - If radicand > 0, TWO possible launch angles, ϕ
    - Smallest ϕ yields earlier time of arrival, t_{hit}
    - Largest ϕ yields later time of arrival, t_{hit}

Target Practice – A Few Examples

V_{init} = 25 m/s

Value of Radicand of $\tan\phi$ equation: **969.31**

Launch angle ϕ: 19.4 deg or 70.6 deg

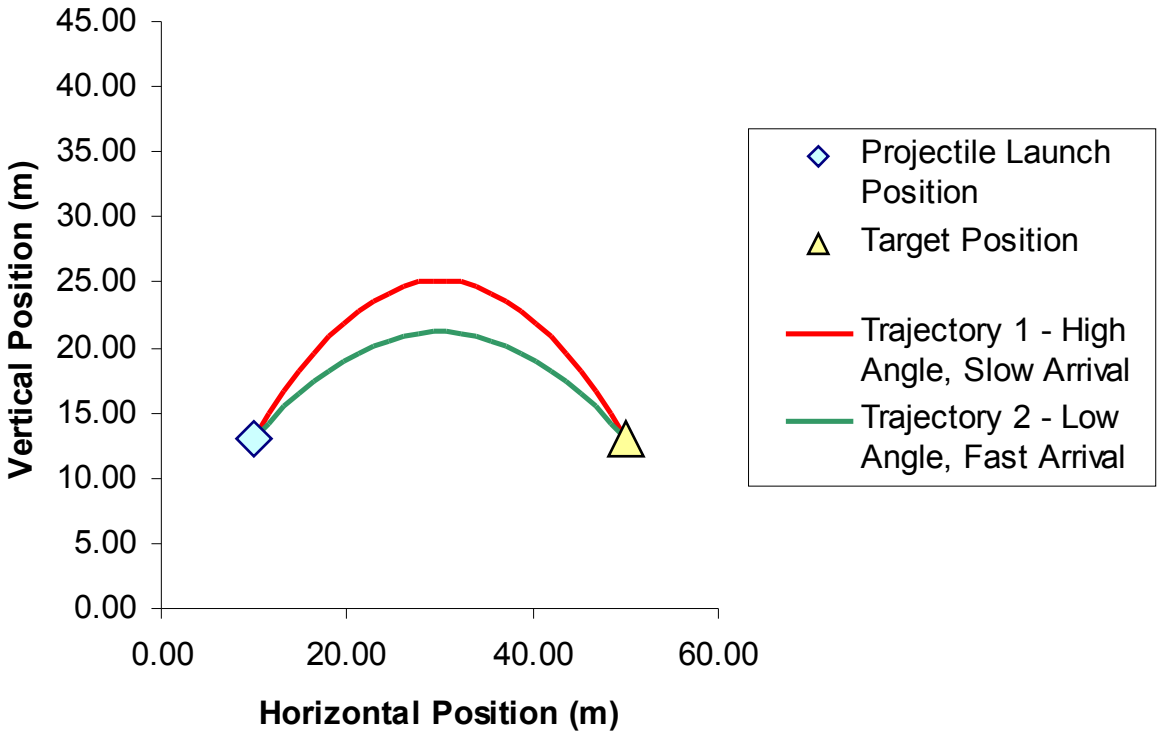


Target Practice – A Few Examples

V_{init} = 20 m/s

Value of Radicand of $\tan\phi$ equation: **60.2**

Launch angle ϕ: 39.4 deg or 50.6 deg

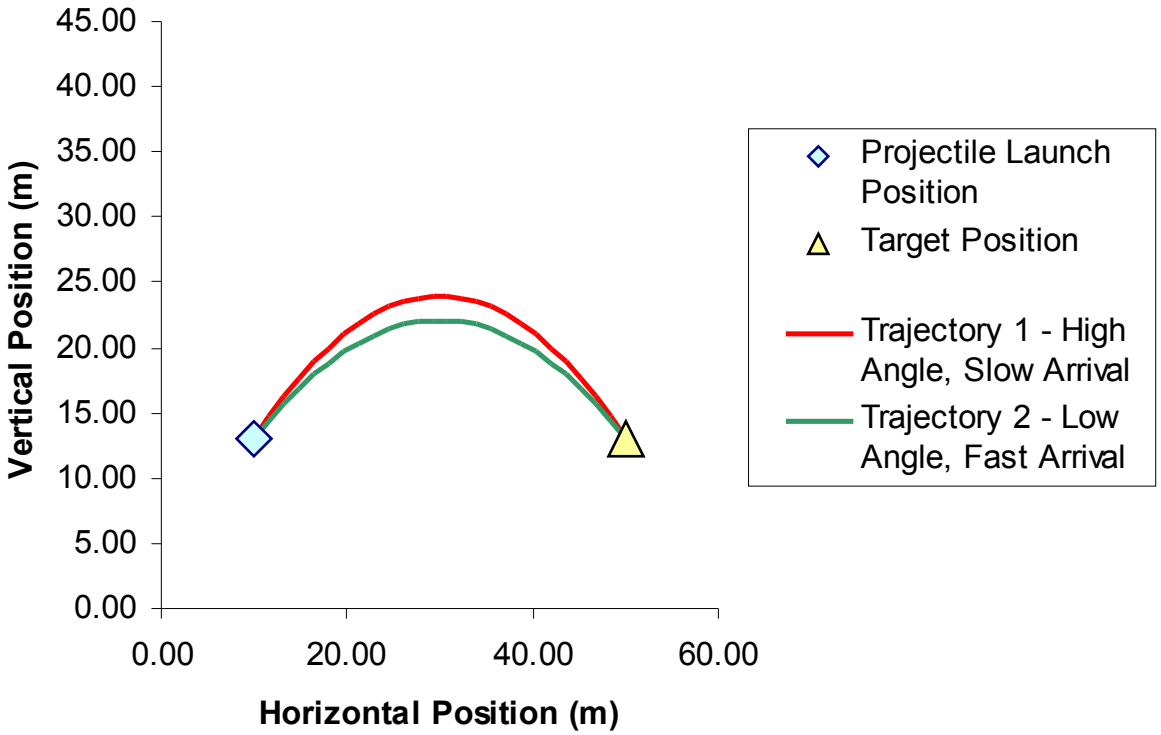


Target Practice – A Few Examples

V_{init} = 19.85 m/s

Value of Radicand of tanϕ equation: **13.2**

Launch angle ϕ: 42.4 deg or 47.6 deg (note convergence)

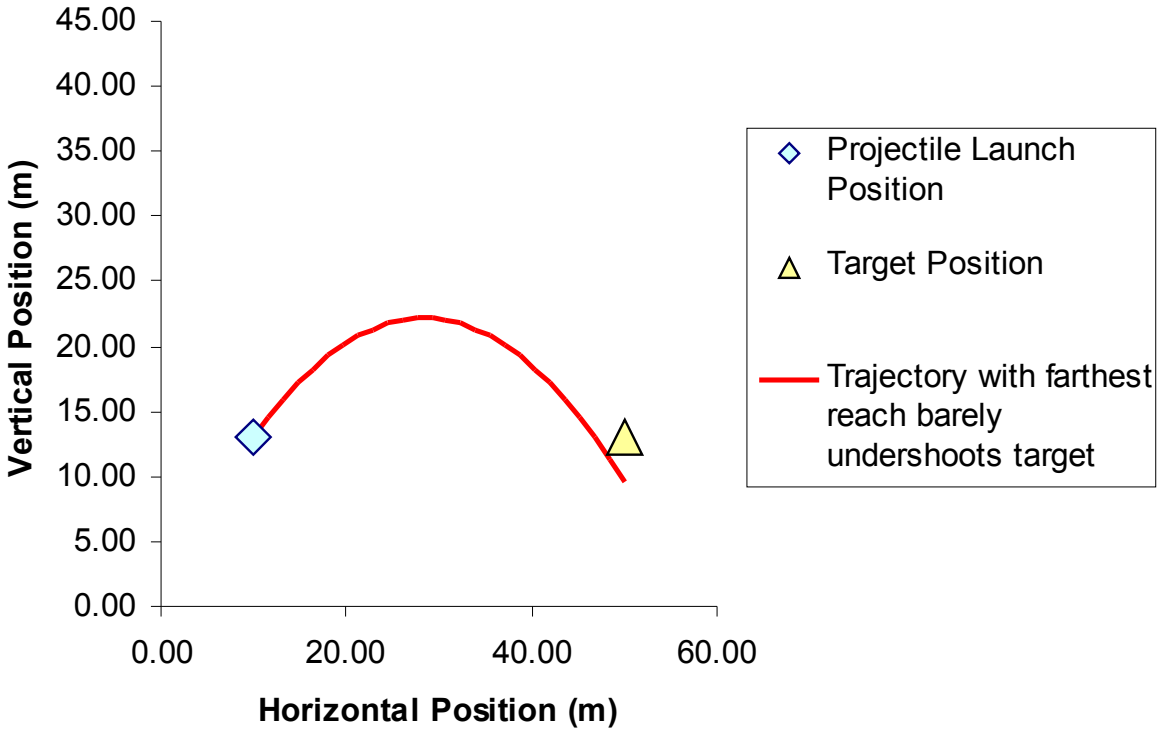


Target Practice – A Few Examples

V_{init} = 19 m/s

Value of Radicand of $\tan\phi$ equation: **-290.4**

Launch angle ϕ: No solution! V_{init} too small to reach target!

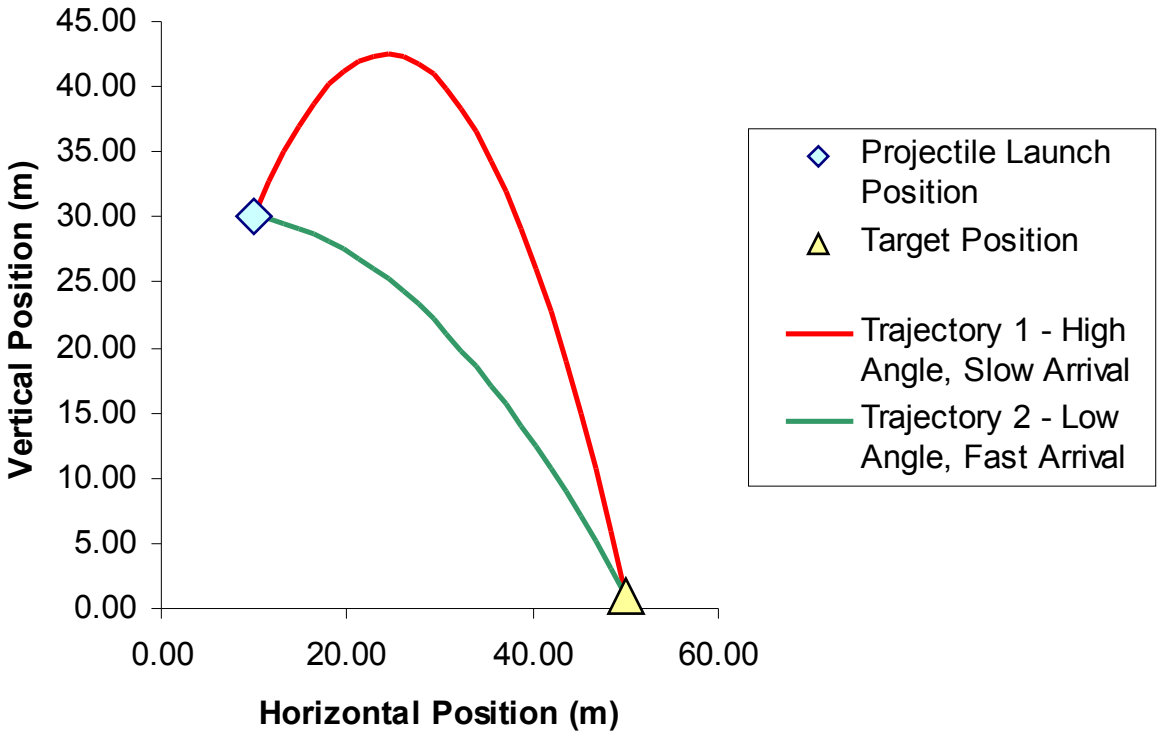


Target Practice – A Few Examples

V_{init} = 18 m/s

Value of Radicand of tanϕ equation: **2063**

Launch angle ϕ: -6.38 deg or 60.4 deg



Target Practice – A Few Examples

V_{init} = 30 m/s

Value of Radicand of tanϕ equation: **668**

Launch angle ϕ: 39.1 deg or 75.2 deg